Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077884

ARTICLE

Structured Random Cycle-Guided Algorithm (SRCA): An Adaptive Metaheuristic Combining Directionally-Guided and Stochastic Search Strategies

Giuseppe Marannano*, Antonino Cirello and Tommaso Ingrassia

Department of Engineering, University of Palermo, Viale delle Scienze, Palermo, Italy

*Corresponding Author: Giuseppe Marannano. Email: giuseppe.marannano@unipa.it

Received: 18 December 2025; Accepted: 26 February 2026; Published: 09 April 2026

ABSTRACT: In response to the growing need for adaptive optimization algorithms capable of handling complex, multimodal, and high-dimensional search spaces, this paper introduces the Structured Random Cycle-guided Algorithm (SRCA). SRCA is not presented as a fundamentally new optimization paradigm, but rather as an architectural synthesis and a unified adaptive framework for dynamic operator selection. Based on a cycle-structured architecture, directional and stochastic search behaviors are dynamically selected at the individual level. The algorithm orchestrates well-established structured movements with a diverse pool of stochastic exploration strategies, enabling a coherent and adaptive balance between exploration and exploitation throughout the optimization process. Unlike traditional metaheuristics that rely on fixed behavioral roles or static movement schemes, SRCA allows each individual to adapt its search strategy based on real-time population feedback, monitored through convergence and dispersion indicators. The performance of SRCA is quantitatively assessed under strictly identical experimental conditions on a comprehensive set of 23 benchmark functions, including multimodal and high-dimensional problems, as well as on six classical constrained engineering design problems. Numerical results demonstrate competitive convergence reliability and robustness across diverse optimization tasks, confirming the effectiveness of the proposed adaptive cycle-based framework.

KEYWORDS: Metaheuristic algorithms; optimization; constrained optimization; benchmark functions; structured random cycle-guided algorithm

1 Introduction

In recent decades, the optimization phase has become a central component in numerous fields, including structural engineering, artificial intelligence, and applied mathematics, serving as a foundational tool for the design and analysis of complex systems [1]. Whether the goal is to minimize energy consumption, maximize structural performance, or fine-tune parameters in machine learning models, the demand for robust and versatile optimization algorithms is increasingly critical. Although classical optimization techniques perform well in smooth, convex, and well-understood problem search spaces, they often struggle when applied to multimodal, non-convex, or poorly defined objective functions. To address these challenges, heuristic and metaheuristic algorithms have gained considerable attention [2]. Inspired by natural processes, biological evolution, and swarm intelligence, these approaches offer flexible frameworks for exploring high-dimensional search spaces without requiring gradient information or convexity assumptions [3]. Their stochastic nature and adaptability make them particularly effective in solving real-world optimization problems characterized by discontinuities, noise, or an undefined mathematical structure. In this regard,

within the field of heuristic optimization, numerous exploration strategies have been developed to balance the dual objectives of global search and local refinement. A common design philosophy among many nature-inspired algorithms is the incorporation of structured movement paradigms, such as linear guidance toward promising solutions, spiral navigation for controlled diversification, and stochastic sampling to ensure global search [4]. Each of these mechanisms fulfills a distinct functional role in the optimization process: linear movements are primarily exploitative, drawing individuals toward known optima; spiral trajectories promote controlled exploration around high-potential regions, maintaining diversity while progressively refining the search; and random sampling enables broad exploration of the solution space, improving the algorithm's ability to escape local optima.

Several well-known metaheuristics embody these principles. For instance, the Group Search Optimizer (GSO) mimics animal group behavior by incorporating a scrounger phase, where individuals move linearly toward a leading member, termed the producer, to intensify search in promising areas [5–7]. The Whale Optimization Algorithm (WOA) [8–11] draws inspiration from the bubble-net feeding behavior of humpback whales, utilizing a spiral-shaped movement to encircle prey (solutions) with both exploitation and mild exploration properties. In contrast, simpler methods like Hill Climbing [12] rely on deterministic and greedy local improvements, which, while efficient in convex search spaces, tend to stagnate in multimodal problems due to their lack of exploratory capacity. To enhance global search capabilities, other algorithms like Cuckoo Search [4,13,14] and those employing Lévy flights introduce heavy-tailed stochastic steps, enabling long-range transitions across the search space. These mechanisms allow the algorithm to jump out of local basins of attraction and reach unexplored regions, significantly boosting exploratory power. Among swarm-based optimization methods, Particle Swarm Optimization (PSO) [15–17] and the Firefly Algorithm (FA) [18–21] have consistently shown strong performance in handling complex search spaces. PSO operates by updating each particle's position using a blend of individual experience and global guidance, facilitating rapid convergence. However, its deterministic update rules may cause premature convergence when faced with fragmented or non-convex fitness search spaces. The Firefly Algorithm employs a light-intensity-based attraction scheme, where individuals are probabilistically drawn toward brighter (i.e., higher-quality) candidates in the population; this model balances intensification and diversification but remains sensitive to user-defined parameters that govern light decay and randomness. In a previous study [22], we introduced adaptive improvements to both GSO and the Firefly Algorithm to enhance their robustness in complex scenarios by promoting diversity and context-aware search. These enhancements highlighted the importance of strategic flexibility and responsiveness to the evolving population dynamics. Recent approaches, such as Hybrid Differential Evolution (DE) [23] and Adaptive PSO [24,25], combine multiple search strategies or adjust parameters based on population feedback. However, these methods often rely on tuning continuous parameters (e.g., inertia weights) or population size, rather than dynamically switching the fundamental search structure itself to match the landscape topology.

Rather than proposing a fundamentally new algorithmic paradigm, we introduce an architectural synthesis of existing concepts named the Structured Random Cycle-guided Algorithm (SRCA). Explicitly, SRCA acts as a unified adaptive framework for operator scheduling that modularly combines structured directionally-guided movements with a diverse pool of stochastic exploration strategies. Specifically, SRCA operates through different directionally-guided movement strategies: linear guidance, spiral navigation (inspired by the encircling behavior modeled in the whale optimization algorithm) and adaptive data-driven movement. These deterministic strategies are complemented by nine randomized exploration mechanisms, including global, local, hybrid (cycle-by-cycle and periodic), standard deviation-guided, memory-based, Gaussian, Lévy flight sampling, and adaptive strategy selection. By coherently integrating these structured and stochastic behaviors, SRCA maintains a balance between robust convergence and continuous exploration

across the optimization process. This contrasts with many existing algorithms that are typically limited to a fixed set of movement rules. For example, the GSO statically assigns individuals to predefined behavioral roles (producer, scrounger, ranger), while the WOA relies almost exclusively on spiral-based movements. Similarly, Cuckoo Search emphasizes long-range transitions through Lévy flights but lacks integrated mechanisms for localized refinement or structured adaptation. In contrast, SRCA enables each individual to explore the search space through a controlled application of diverse stochastic strategies, selected adaptively according to the search context and on real-time population feedback. By monitoring convergence and variable dispersion, SRCA adapts its search focus, favoring exploration initially and shifting toward exploitation as convergence improves, thus maintaining a balanced search and mitigating premature convergence.

To accurately reflect the methodological positioning of this work, the main contributions are framed in terms of architectural synthesis rather than conceptual novelty: (i) the introduction of a unified adaptive framework for operator scheduling, where directional and stochastic operators are dynamically selected at the individual level; (ii) the implementation of an adaptive control mechanism based on population-level convergence and dispersion indicators; (iii) the rigorous quantitative validation of the framework's internal architecture through a dedicated ablation study; and (iv) the demonstration of its highly competitive performance on both mathematical benchmarks and complex structural engineering problems under strictly normalized computational budgets.

From an artificial intelligence perspective, SRCA functions as an adaptive decision-making system governed by procedural rules. It is necessary to explicitly distinguish this approach from other adaptive paradigms. Unlike *parameter control* methods (which tune numerical coefficients like inertia weights) or *operator probability adaptation* (which updates roulette-wheel selection chances based on historical success), SRCA employs a direct, rule-based switching mechanism that completely alters the structural search behavior based on current population metrics. Furthermore, unlike *hyper-heuristics*, which often employ machine learning to generate heuristics over time, SRCA's adaptivity relies entirely on a computationally lightweight, threshold-driven logic.

To objectively assess the performance of the proposed algorithm, SRCA is tested against state-of-the-art methods under strictly identical experimental conditions (normalized by maximum function evaluations) on a comprehensive set of multimodal and high-dimensional benchmark functions, along with a selection of six structural optimization problems. Compared to well-known metaheuristics, the algorithm proves particularly effective in maintaining consistent convergence behavior and robustness across diverse optimization tasks.

2 Methodology

As already mentioned in the introduction section, SRCA operates on a population of candidate solutions and evolves them over multiple iterations, integrating a two-phase process: a directional movement phase and a stochastic generation phase. The directional movement is primarily designed to guide individuals toward promising areas of the search space. By default, this movement is directed toward the current best solution x_{best} in the population. However, the framework also includes the possibility to activate an elite-guided mode, in which movement is influenced by the top x_{elite} best solutions instead of a single one. In this case, the algorithm computes a centroid of the elite set ($\overline{x}_{\text{elite}}$) and allows individuals to move toward this collective reference, enhancing the algorithm's capacity to exploit multiple high-quality regions while reducing the risk of premature convergence to a single attractor. The second phase involves the stochastic generation of new candidates using a structured random strategy selector that alternates between different exploration heuristics, thereby ensuring both local refinement and global coverage of the search space.

2.1 Initialization

At the beginning of the algorithm, a population array $P \in \mathbb{R}^{N\times(1+d)}$ is initialized, where N denotes the number of individuals in the population and d is the number of decision variables of the optimization problem. Each row of P corresponds to a candidate solution: the first column stores the fitness value f_i, while the remaining d columns contain the decision variables $x_{i,1}, x_{i,2}, \ldots, x_{i,d}$. The initial values of these variables are generated using a uniform random distribution within the predefined lower and upper bounds of the search space. More formally, each individual can be expressed as:

$$P_i = \left(f_i, x_{i,1}, x_{i,2}, \ldots, x_{i,d}\right) \tag{1}$$

where f_i is the fitness associated with the i-th individual and $x_{i,j}$ represents the j-th variable of that solution. The array is sorted in ascending order based on fitness, so that the first row P_1 corresponds to the current best solution.

2.2 Directionally-Guided Exploitation Strategies

All individuals except the best one are updated through a movement mechanism that drives them closer to the current best solutions. These directionally-guided moves are primarily responsible for the exploitation phase, providing a structured path toward optimality compared to pure random exploration. Furthermore, the framework allows new directional operators to be added to the move pool seamlessly due to its modular architecture. Four distinct strategies have been implemented and tested. In the following formulations, the movement is described with respect to x_{best}, but it can equivalently be directed toward $\overline{x}_{\text{elite}}$, the centroid of the top-performing individuals.

Linear Movement Each variable $x_{i,j}$ is updated using a weighted vector pointing toward the best-known solution $x_{\text{best},j}$ (see Eq. (2), where t denotes the current iteration or analysis step).

$$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + r \cdot \left(x_{\text{best},j}^{(t)} - x_{i,j}^{(t)}\right) \tag{2}$$

where $r \in [-1, 1]$ is a random number in the range $[-1, 1]$.

Spiral Movement Inspired by natural spiral navigation patterns [9], this strategy updates each variable by combining both radial and angular components, enabling the solution to follow a helical trajectory around a target point (see Eq. (3)).

$$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + \rho \cdot \cos(\theta) \cdot \left(x_{\text{best},j}^{(t)} - x_{i,j}^{(t)}\right) + \rho \cdot \sin(\theta) \cdot \xi \tag{3}$$

where $\theta \in [0, 2\pi]$ is a random angle (which is applied component-wise for highly multidimensional navigation) determining the direction of rotation, ρ is a decaying amplitude that decreases linearly with the iteration count to ensure convergence, and $\xi \in [-0.5, 0.5]$ is a uniformly distributed random scalar that introduces controlled perturbation along the spiral path. The maximum absolute value is 0.5, so ξ acts as a moderate scaling factor; it is sufficient to introduce diversification, while preserving convergence stability.

Adaptive Data-Driven Movement. This is a newly introduced strategy designed to enhance the responsiveness of the algorithm to the evolving state of the population. Instead of applying a fixed movement rule, each variable is updated through a directionally biased mechanism driven by real-time population statistics. The update formulation integrates two complementary indicators:

- C_t: the fitness gap between the worst and the best individuals in the current population;
- $\bar{\sigma}_t$: the average standard deviation across all decision variables.

The procedure involves the computation of two dynamic coefficients, α and β, based on C_t and $\bar{\sigma}_t$, respectively.

$$C_t = f_{\text{worst}}^{(t)} - f_{\text{best}}^{(t)}, \qquad \alpha = 1 - \frac{t}{T}, \qquad \beta = \frac{\bar{\sigma}_t}{x_{\max,j}^{(t)} - x_{\min,j}^{(t)}} \tag{4}$$

where t is the current iteration and T is the maximum number of iterations. The coefficient α controls the exploitation pressure, decreasing linearly over time, while β scales the exploratory component based on the diversity of the population. The average standard deviation $\bar{\sigma}_t$ is computed by first evaluating the standard deviation $\sigma_j^{(t)}$ of each decision variable x_j across the current population, and then averaging over all d variables:

$$\bar{\sigma}_t = \frac{1}{d} \sum_{j=1}^{d} \sigma_j^{(t)} \tag{5}$$

This metric reflects the overall dispersion of the population in the decision space at iteration t, and enables the adaptive modulation of the search intensity. Each variable $x_{i,j}$ is updated according to the following equation:

$$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + \alpha \cdot r \cdot \left(x_{\text{best},j}^{(t)} - x_{i,j}^{(t)} \right) + \beta \cdot r \tag{6}$$

where $r \in [-1, 1]$ is a uniformly distributed random scalar. The first term directs the solution toward the current best individual with strength modulated by α, while the second introduces a diversity-preserving perturbation, scaled by β. Although this movement rule is structurally focused on exploiting promising regions, the inclusion of the uniformly random scalar r classifies it as a directionally-guided stochastic update rather than a purely deterministic one. This balances the algorithm's capacity to intensify search near optima and maintain sufficient exploration to escape local traps.

Although C_t does not appear directly in the update rule of Eq. (6), it governs the dynamic adjustment of the parameters α and β as described below. In fact, to further enhance adaptability, a conditional adjustment mechanism is introduced in the algorithm based on the convergence indicator C_t. This mechanism alters the values of α and β according to three distinct scenarios:

- If $C_t \leq 0.01$, diversification mechanisms are activated by increasing the exploratory component β, which is doubled ($\beta \to 2 \cdot \beta$). The convergence factor α remains unchanged and is computed as $\alpha = 1 - t/T$. This configuration enhances the algorithm's ability to escape premature convergence.
- If $0.01 < C_t \leq 0.1$, a controlled exploitation strategy is applied: the convergence factor α is reduced by 20% ($\alpha \to 0.8 \cdot \alpha$), and β is increased by 20% ($\beta \to 1.2 \cdot \beta$). This adjustment favors fine-tuning near promising regions.
- If $C_t > 0.1$, the population is considered sufficiently diverse. The default update rule is applied, with $\alpha = 1 - t/T$ and β computed as in Eq. (4). No corrective adjustment is introduced.

To transition from empirical justification to a formally robust configuration, a targeted sensitivity analysis was conducted on the adjustment factors (e.g., the α decay and β scaling) and the C_t threshold values. By evaluating diverse search landscapes (unimodal, multimodal, and highly constrained), the analysis verified that the C_t thresholds of 0.01 and 0.1 provide the most reliable trigger points to shift between exploitation and exploration, while the scaling bounds prevent divergent oscillatory behavior. This rigorous robustness analysis ensures that the adaptive rules are not merely empirical estimates, but structurally sound control mechanisms for maintaining population diversity. These values ensured a stable balance

between exploration and exploitation across heterogeneous test cases. More aggressive adjustments resulted in unstable oscillatory behavior, whereas milder variations had a negligible impact on convergence dynamics. To further prevent instability near the threshold conditions, the control rules are evaluated at each iteration, while the effects of parameter updates are naturally smoothed by the progressive decay of α and the statistical averaging embedded in β.

Directional Ensemble Movement. In the work, a fourth strategy aims to combine the strengths of the previous mechanisms by integrating them into a unified ensemble update. Rather than relying on a single movement model, this approach blends the linear, spiral, and adaptive components into a single formulation:

$$x_{i,j}^{(t+1)} = \omega_1 \cdot x_{i,j}^{(\text{linear})} + \omega_2 \cdot x_{i,j}^{(\text{spiral})} + \omega_3 \cdot x_{i,j}^{(\text{adaptive})} \tag{7}$$

By combining multiple directional components, the directional ensemble strategy enables a richer and more balanced navigation of the search space. This formulation mitigates the weaknesses of any single method and increases robustness across different search space topologies. The weights can either be fixed or adaptively adjusted according to real-time feedback from the search dynamics. To ensure a robust baseline performance without problem-specific tuning, a limited sensitivity analysis was conducted on a representative subset of structurally distinct functions (F1 for unimodality, F10 for multimodality, and F22 for asymmetrical local traps). The analysis evaluated variations in C_t thresholds, ensemble weights ($\omega_1, \omega_2, \omega_3$), and population size distribution. Results confirmed that a balanced deterministic-to-stochastic ratio and an ensemble weighting skewed toward the adaptive data-driven movement ($\omega_1 = 0.2$, $\omega_2 = 0.3$, $\omega_3 = 0.5$) yield the most stable convergence across diverse topologies. To explicitly substantiate this, Table 1 reports a summary of the sensitivity analysis, demonstrating how variations in these hyper-parameters impact the mean convergence.

Table 1: Sensitivity analysis of internal parameters and ensemble weights (Mean fitness over 25 runs).

Configuration	C_t Thresholds	Weights ($\omega_{1,2,3}$)	N_{det}/N_{off}	F1	F10	F22
Default (Proposed)	(0.01, 0.1)	(0.2, 0.3, 0.5)	100/100	1.63E-04	9.76E-03	−10.4029
Relaxed Thresholds	(0.05, 0.2)	(0.2, 0.3, 0.5)	100/100	2.14E-01	7.74E-01	−9.2201
Even Weights	(0.01, 0.1)	(0.33, 0.33, 0.34)	100/100	5.33E-02	8.82E-02	−9.8341
Exploration-Heavy	(0.01, 0.1)	(0.2, 0.3, 0.5)	50/150	4.09E-01	8.83E-01	−8.4011

While this static configuration demonstrated strong generalization capabilities and was adopted for all subsequent experiments, we explicitly acknowledge that an exhaustive, problem-specific tuning of the C_t thresholds and ensemble weights could potentially unlock further performance improvements for specific optimization tasks.

2.3 Stochastic Exploration Strategies

To maintain diversity and exploration, a second set of solutions is generated using random movements based on predefined or adaptive strategies. These new individuals are appended to the population P and evaluated using the objective function. The algorithm supports nine random movement strategies, selected via a control parameter. Moreover, if required and in automatic mode, the algorithm permits a fully adaptive mode (strategy 9), where the choice of strategy depends on the convergence and diversity of the population: when the population is stagnant, global exploration is favored; when it's highly dispersed, localized refinement is applied. In the work, the implemented strategies are:

1. Global Random Sampling (GRS): Uniform random variables in the search space;
2. Local Random Sampling (LRS): Sampling around the best solution with shrinking amplitude;
3. Cycle-by-Cycle Hybrid: Alternating global and local at each iteration;
4. Periodic Hybrid: Periodic switching between global and local every three cycles;
5. Standard Deviation-Guided Sampling: Perturbation proportional to the inverse of variable-wise standard deviation;
6. Memory-Based Sampling: Centered around historical best values;
7. Gaussian Perturbation: Centered on the best solution with added Gaussian noise;
8. Lévy Flight: Long-tailed random steps for global jumps;
9. Adaptive Strategy Selection: Dynamic control of search behavior.

Each strategy, whose underlying mechanism is discussed in detail in Section 3, aims to address a specific trade-off between diversification and intensification, enabling a coherent balance between global exploration and local refinement.

2.4 Fitness Evaluation, Population Update, and Probabilistic Convergence

After each distinct phase (firstly the directional movement, and subsequently the stochastic exploration), the algorithm merges the newly generated candidate solutions with the current population. The extended array is immediately re-sorted by fitness, and only the best N individuals are retained. This two-stage intra-generational elitism accelerates convergence by ensuring that the stochastic phase strictly operates on the refined outcomes of the deterministic phase. This strict retention of the top-performing individuals ensures that the sequence of best solutions is monotonically non-increasing in terms of fitness, i.e., $f(x_{\text{best}}^{(t+1)}) \leq f(x_{\text{best}}^{(t)})$. This elitist property is fundamental for theoretical stability.

Furthermore, the algorithmic architecture provides a formal probabilistic guarantee of global convergence. According to stochastic search theory, a metaheuristic converges to the global optimum in probability if two conditions are met: (1) the process is elitist, and (2) the probability of generating a solution in any subset of the search space with a positive measure (including the global optimum region) is strictly greater than zero at every iteration. In SRCA, condition (1) is satisfied by the aforementioned sorting and truncation mechanism. Condition (2) is mathematically guaranteed by the stochastic exploration phase: the continuous availability of the Global Random Sampling (Strategy 1) and the heavy-tailed Lévy flight (Strategy 8) ensures that every point in the domain S has a probability $P(x \in S) > 0$ of being sampled. Consequently, as the number of evaluations approaches infinity, the probability of finding the global optimum approaches one, providing a rigorous probabilistic stability guarantee for the SRCA framework.

2.5 Iteration and Termination

The entire process, movement, exploration, fitness evaluation, and sorting, is repeated for a fixed number of iterations or until a convergence criterion is met (e.g., negligible improvement in fitness). Throughout the process, the best solution found so far is tracked and returned at the end of the optimization.

2.6 Computational Complexity and Cost

A critical aspect of any metaheuristic is its computational complexity. The time complexity of SRCA is primarily determined by the objective function evaluation, the sorting mechanism, and the update rules. Let N be the population size, D the problem dimension, and $MaxIt$ the maximum number of iterations.

The complexity of the initialization is $O(N \cdot D)$. In the main loop, SRCA performs two major phases:

1. *Deterministic Update:* Requires $O(N \cdot D)$ operations.
2. *Stochastic Exploration:* Generates an additional set of N candidates, requiring another $O(N \cdot D)$ operations and N function evaluations.

Consequently, SRCA evaluates $2N$ candidate solutions per iteration. To ensure a strictly fair and unbiased comparison across all tested algorithms, the evaluation budget is rigorously governed by a predefined maximum number of function evaluations (MaxFEs) rather than a fixed iteration count. Therefore, while SRCA processes more candidates per cycle compared to algorithms like PSO or GSO, the total computational budget remains identical across all comparators. This rigorous normalization ensures that any performance superiority is exclusively attributable to the algorithmic efficiency and the structural integration of directionally-guided and stochastic behaviors.

To synthesize the overall execution flow and facilitate full reproducibility, the step-by-step procedure of the proposed framework is detailed in Algorithm 1.

Algorithm 1: Structured random cycle-guided algorithm (SRCA)

Input: Population size N, Dimension d, Bounds $[x_{\min}, x_{\max}]$, MaxFEs
Initialize population P with N randomly generated individuals
Evaluate fitness $f(x)$ for all $x \in P$; $FEs \leftarrow N$
Sort P in ascending order of fitness; identify the global best x_{best}
while $FEs < MaxFEs$ **do**
    Calculate convergence indicator $C_t = f_{\text{worst}} - f_{\text{best}}$ and diversity $\bar{\sigma}_t$ (Eq. (5))
    Compute and conditionally adjust α and β based on C_t thresholds
    // Phase 1: Directionally-guided Exploitation
    for $i = 2$ **to** N **do**
        Compute x_{linear}, x_{spiral}, and x_{adaptive} guided by x_{best}
        Combine using ensemble weights: $x_i' = \omega_1 x_{\text{linear}} + \omega_2 x_{\text{spiral}} + \omega_3 x_{\text{adaptive}}$
        Apply boundary clipping to x_i'
    end for
    Evaluate Phase 1 individuals; $FEs \leftarrow FEs + (N-1)$
    Merge, sort, and retain the best N individuals; update x_{best}
    // Phase 2: Stochastic Exploration
    for $i = 1$ **to** N **do**
        Select strategy index $s \in \{1, \ldots, 8\}$ using Adaptive Strategy 9 (based on C_t, $\bar{\sigma}_t$)
        Generate x_i'' using the selected stochastic strategy s
        Apply boundary clipping to x_i''
    end for
    Evaluate Phase 2 individuals; $FEs \leftarrow FEs + N$
    Merge, sort, and retain the best N individuals; update x_{best}
end while
Output: Global best solution x_{best} and its optimal fitness f_{best}

3 Description of the Random Strategies

This section details the nine implemented exploration strategies.

Strategy 1-Global Random Sampling (GRS). As the fundamental form of stochastic exploration, GRS generates each decision variable $x_{i,j}$ independently using a uniform probability distribution across the admissible range:

$$x_{i,j}^{(t+1)} \in \left[x_{i,j}^{\min}, x_{i,j}^{\max}\right] \tag{8}$$

where $x_{i,j}^{\min}$ and $x_{i,j}^{\max}$ define the lower and upper bounds. GRS maximizes diversity by encouraging independent exploration unconstrained by the population state, making it effective for escaping local optima in early search stages.

Strategy 2-Local Random Sampling (LRS). LRS focuses on exploitation by generating solutions within a shrinking neighborhood of the current best individual. The update rule is defined as:

$$x_{i,j}^{(t+1)} = x_{best,j}^{(t)} + r \cdot A_k \tag{9}$$

where $r \in [-1, 1]$ is a uniform random number, and A_k is the local search amplitude at iteration k:

$$A_k = l_{max} - \delta_l \cdot (k-1) \tag{10}$$

Here, l_{max} is the initial radius and δ_l is a fixed decrement. This formulation ensures the search radius contracts over time, allowing adaptive refinement around promising regions.

Strategy 3-Cycle-by-Cycle Hybrid Sampling. This strategy alternates between global and local sampling at each iteration based on the cycle number k, balancing diversification and intensification:

$$x_{i,j}^{(t+1)} = \begin{cases} x_{i,j}^{(t+1)} \in \left[x_{i,j}^{\min}, x_{i,j}^{\max}\right], & \text{if } k \bmod 2 = 0 \quad \text{(GRS)} \\ x_{i,j}^{(t+1)} = x_{best,j}^{(t)} + r \cdot A_k, & \text{if } k \bmod 2 = 1 \quad \text{(LRS)} \end{cases} \tag{11}$$

Strategy 4-Periodic Hybrid Sampling. Unlike the rapid switching of Strategy 3, this method introduces persistent phases of exploration or exploitation by alternating behavior every m iterations:

$$x_{i,j}^{(t+1)} = \begin{cases} x_{i,j}^{(t+1)} \in \left[x_{i,j}^{\min}, x_{i,j}^{\max}\right], & \text{if } \left\lfloor \frac{k}{m} \right\rfloor \bmod 2 = 0 \\ x_{i,j}^{(t+1)} = x_{best,j}^{(t)} + r \cdot A_k, & \text{otherwise} \end{cases} \tag{12}$$

This periodic modulation allows for more stable transitions, benefiting search in multimodal landscapes.

Strategy 5-Standard Deviation-Guided Sampling. This strategy uses population statistics to modulate perturbations based on variable spread. For each variable $x_{i,j}$, the update rule is:

$$x_{i,j}^{(t+1)} = x_{best,j}^{(t)} + \eta \cdot r \cdot \left(1 + \frac{1}{\max(\sigma_j, \epsilon)}\right) \tag{13}$$

where $r \in [-1, 1]$, σ_j is the standard deviation of variable j, and ϵ is a small constant. Here, perturbations are inversely proportional to population spread: low variance triggers larger steps to explore neglected directions, while high variance induces smaller steps to preserve stability.

Strategy 6-Memory-Based Sampling. Instead of relying solely on the current best, this strategy utilizes an archive of historical elite solutions. The new solution is generated as:

$$x_{i,j}^{(t+1)} = \bar{x}_j + \xi \tag{14}$$

where $\bar{x}_j$ is the mean value of the j-th variable computed across all archived best solutions (Eq. (15)), and $\xi \in [-0.5, 0.5]$ is a random perturbation.

$$\bar{x}_j = \frac{1}{M} \sum_{m=1}^{M} x_{m,j} \tag{15}$$

This method exploits evolutionary memory to revisit promising regions that may not be adjacent to the current global best.

Strategy 7-Gaussian Perturbation. This strategy performs probabilistic local search by perturbing the best solution with Gaussian noise to explore its immediate neighborhood:

$$x_{i,j}^{(t+1)} = x_{best,j}^{(t)} + \mathcal{N}(0, \sigma^2) \tag{16}$$

where $\mathcal{N}(0, \sigma^2)$ is a sample from a normal distribution. Gaussian noise naturally favors small refinements while permitting occasional larger deviations due to the distribution's tails.

Strategy 8-Lévy Flight. Inspired by natural foraging behavior, Lévy flights introduce heavy-tailed random steps to enable long-distance jumps [26]. The update rule is:

$$x_{i,j}^{(t+1)} = x_{best,j}^{(t)} + \text{Lévy}(\beta) \tag{17}$$

where $\text{Lévy}(\beta)$ is drawn from a Lévy distribution (Eq. (18)) with index $\beta = 1.5$.

$$\text{Lévy}(\beta) = \frac{u}{|v|^{1/\beta}} \tag{18}$$

with σ_u computed as per Eq. (19).

$$\sigma_u = \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi\beta/2)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{(\beta-1)/2}} \right)^{1/\beta} \tag{19}$$

This scale-free movement is particularly effective for escaping local minima in sparse search spaces.

Strategy 9-Adaptive Strategy Selection. This fully adaptive mode dynamically selects the exploration strategy based on real-time population feedback, guided by convergence C_t and dispersion $\bar{\sigma}_t$:

- Stagnation detection: If $C_t < 0.01$ and $\bar{\sigma}_t < 0.1$, apply Strategy 1 (Global Sampling).
- Low convergence: If $0.01 < C_t < 0.1$, apply Strategy 5 (Std Dev Sampling).
- Continuous injection: Apply Strategy 8 (Lévy Flight) to a uniformly distributed subset of the population (e.g., 20%) at each iteration to guarantee persistent heavy-tailed diversification.
- High dispersion: If $\bar{\sigma}_t > 5$, apply Strategy 2 (Local Sampling).
- Default: Randomly select from $\{1, \ldots, 8\}$.

This mechanism provides context-sensitive control without the computational overhead of training a learning model. As demonstrated by the sensitivity analysis summarized in Section 2.2, the selected switching conditions represent robust operational boundaries verified to prevent premature stagnation across varied landscapes. By dynamically activating or deactivating specific search strategies based on their immediate impact on population metrics, Strategy 9 inherently performs continuous, run-time operator evaluation. While this dynamic behavior offers valuable insights into operator efficacy during the search process, it complements rather than replaces formal structural validation. Therefore, a dedicated static

ablation study is explicitly provided in Section 4.3 to quantitatively assess and empirically justify the independent contribution of each algorithmic module.

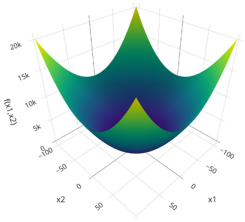
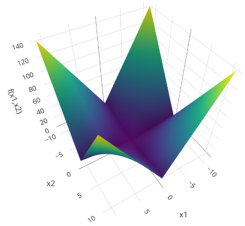
4 Benchmark Verification

4.1 Performance Evaluation on Benchmark Functions

The algorithm SRCA was fully developed using Python 3.12 environment. While platforms such as MATLAB are frequently used for numerical optimization due to their extensive libraries and integrated visualization tools, the choice of Python was driven by its open-source nature, rich ecosystem, and flexibility. To ensure rigorous comparative conditions across all tested algorithms, the termination criterion was defined strictly by the maximum number of function evaluations (MaxFEs) rather than a fixed number of iterations. The MaxFEs budget was dynamically determined based on the problem's dimensionality (D): 100,000 for $D \leq 10$; 200,000 for $D \leq 30$; 400,000 for $D \leq 50$; 800,000 for $D \leq 150$; and 1,000,000 otherwise. For the SRCA algorithm, the population size was maintained at 100 individuals, with an additional 100 randomly generated candidates evaluated at each iteration, operating strictly within the assigned MaxFEs limit. All comparator algorithms were rigorously evaluated under these exact same MaxFEs constraints over 25 independent runs to guarantee a completely normalized, statistically robust, and equitable performance assessment.

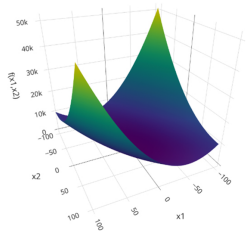
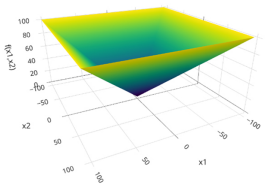
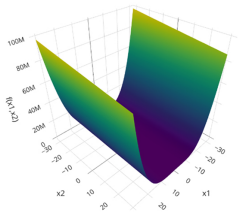
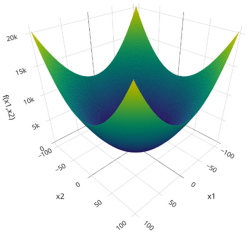
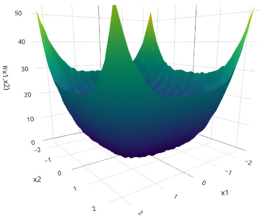
To evaluate the performance and robustness of the SRCA algorithm, a comprehensive suite of benchmark functions was employed [9,27,28]. These functions (named F1-F23), widely adopted in the literature, are used to assess the global exploration capabilities, convergence speed, and resilience to local optima across various problem types, including unimodal, multimodal, separable, and non-separable search spaces. Table 2 reports the benchmark functions, including their respective domains and known global minima. For each function, a 3D plot in two variables is also included.

Table 2: Benchmark functions used in the study.

Benchmark Function	3D Plot
F1–Sphere Function $f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$ num. variables: 30 range: $[-100; 100]$ fmin: 0	
F2–Schwefel 2.22 Function $f(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$ num. variables: 30 range: $[-10; 10]$ fmin: 0	

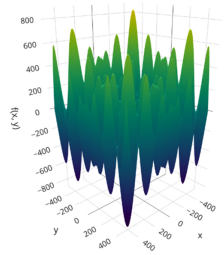
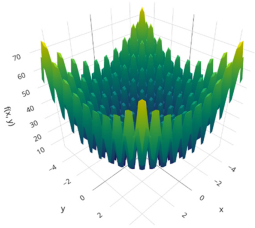
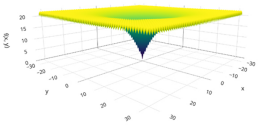
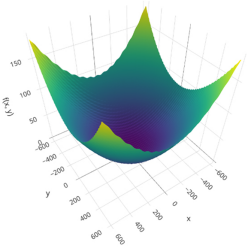
(Continued)

Table 2 (continued)

Benchmark Function	3D Plot
F3–Schwefel 1.2 Function $f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$ num. variables: 30 range: $[-100; 100]$ fmin: 0	
F4–Schwefel 2.21 Function $f(\mathbf{x}) = \max_i \lvert x_i \rvert$ num. variables: 30 range: $[-100; 100]$ fmin: 0	
F5–Rosenbrock Function $f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$ num. variables: 30 range: $[-30; 30]$ fmin: 0	
F6–Step Function $f(\mathbf{x}) = \sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor^2$ num. variables: 30 range: $[-100; 100]$ fmin: 0	
F7–Quartic with Noise Function $f(\mathbf{x}) = \sum_{i=1}^{n} x_i^4 + \text{random}[0, 1]$ num. variables: 30 range: $[-1.28; 1.28]$ fmin: ~ 0	

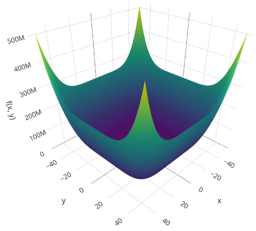
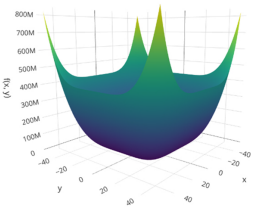
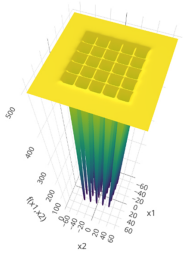
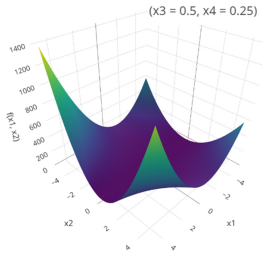
(Continued)

Table 2 (continued)

Benchmark Function	3D Plot
F8–Schwefel Function $f(\mathbf{x}) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$ num. variables: 30 range: $[-500; 500]$ fmin: −12569.5	
F9–Rastrigin Function $f(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$ num. variables: 30 range: $[-5.12; 5.12]$ fmin: 0	
F10–Ackley Function $f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$ $+20 + e$ num. variables: 30 range: $[-32; 32]$ fmin: 0	
F11–Griewank Function $f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$ num. variables: 30 range: $[-600; 600]$ fmin: 0	

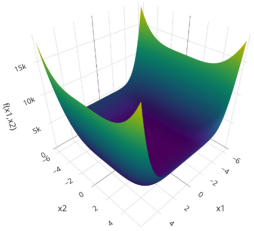
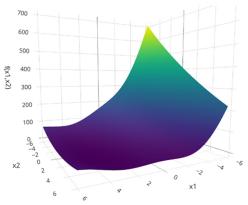
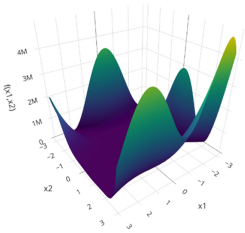
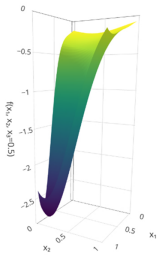
(Continued)

Table 2 (continued)

Benchmark Function	3D Plot
F12–Penalized Function #1 $f(\mathbf{x}) = \frac{\pi}{n}\Big[10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left(1 + 10\sin^2(\pi y_{i+1})\right) + (y_n - 1)^2\Big] + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4},\ u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ k(-x_i - a)^m & x_i < -a \\ 0 & \text{otherwise} \end{cases}$ num. variables: 30 range: $[-50; 50]$ fmin: 0	
F13–Penalized Function #2 $f(\mathbf{x}) = 0.1\Big[\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2\left(1 + \sin^2(3\pi x_{i+1})\right) + (x_n - 1)^2\left(1 + \sin^2(2\pi x_n)\right)\Big] + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ k(-x_i - a)^m & x_i < -a \\ 0 & \text{otherwise} \end{cases}$ num. variables: 30 range: $[-50; 50]$ fmin: 0	
F14–Shekel's Foxholes Function $f(x_1, x_2) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + (x_1 - a_{1j})^6 + (x_2 - a_{2j})^6}\right]^{-1}$ with $a = \begin{bmatrix} -32 & -16 & 0 & \dots & 32 \\ -32 & -32 & -32 & \dots & 32 \end{bmatrix}$ num. variables: 2 range: $[-65.536; 65.536]$ fmin: ≈ 1	
F15–Kowalik's Function $f(x_1, x_2, x_3, x_4) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$ where $\begin{cases} a = \{0.1957, 0.1947, \dots, 0.0246\} \\ b = \{0.25, 0.5, \dots, 16.0\} \end{cases}$ num. variables: 4 range: $[-5; 5]$ fmin: $\approx 3.07 \times 10^{-4}$	

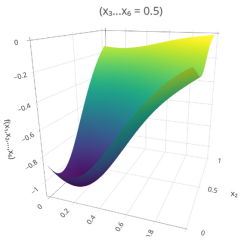
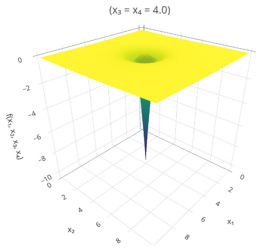
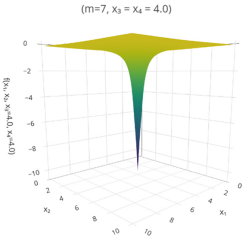
(Continued)

Table 2 (continued)

Benchmark Function	3D Plot
F16–Six-Hump Camel Back Function $f(x_1, x_2) = \left(4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3}\right) + x_1x_2 + (-4 + 4x_2^2)x_2^2$ num. variables: 2 range: $[-5; 5]$ fmin: ≈ -1.0316	
F17–Branin Function $f(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ num. variables: 2 range: $[-5; 5]$ fmin: ≈ 0.3979	
F18–Goldstein-Price Function $f(x_1, x_2) =$ $\left[1 + (x_1 + x_2 + 1)^2 \cdot \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right]$ $\times \left[30 + (2x_1 - 3x_2)^2 \cdot \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$ num. variables: 2 range: $[-2; 2]$ fmin: 3	
F19–Hartmann 3D Function $f(\mathbf{x}) = -\sum_{i=1}^{4} c_i \cdot \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$ where: $\mathbf{c} = [1.0,\ 1.2,\ 3.0,\ 3.2]$ $\mathbf{A} = \begin{bmatrix} 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \\ 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \end{bmatrix}$ $\mathbf{P} = \begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.0381 & 0.5743 & 0.8828 \end{bmatrix}$ num. variables: 3 range: $[0; 1]$ fmin: ≈ -3.86	

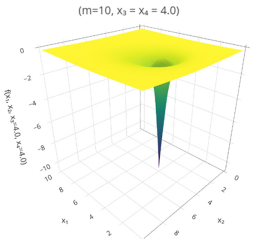
(Continued)

Table 2 (continued)

Benchmark Function	3D Plot
F20–Hartmann 6D Function $f(\mathbf{x}) = -\sum_{i=1}^{4} c_i \cdot \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$ $\mathbf{c} = [1.0,\ 1.2,\ 3.0,\ 3.2]$ $\mathbf{A} = \begin{bmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{bmatrix}$ $\mathbf{P} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$ num. variables: 6 range: $[0;1]$ fmin: ≈ -3.32	
F21–Shekel-5 Function $f(\mathbf{x}) = -\sum_{i=1}^{5} \frac{1}{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i}$ $\mathbf{c} = [0.1,\ 0.2,\ 0.2,\ 0.4,\ 0.4]$ $\mathbf{A} = \begin{bmatrix} 4.0 & 4.0 & 4.0 & 4.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 8.0 & 8.0 & 8.0 & 8.0 \\ 6.0 & 6.0 & 6.0 & 6.0 \\ 3.0 & 7.0 & 3.0 & 7.0 \end{bmatrix}$ num. variables: 4 range: $[0;10]$ fmin: ≈ -10.15	
F22–Shekel-7 Function $f(\mathbf{x}) = -\sum_{i=1}^{7} \frac{1}{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i}$ $\mathbf{c} = [0.1,\ 0.2,\ 0.2,\ 0.4,\ 0.4,\ 0.6,\ 0.3]$ $\mathbf{A} = \begin{bmatrix} 4.0 & 4.0 & 4.0 & 4.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 8.0 & 8.0 & 8.0 & 8.0 \\ 6.0 & 6.0 & 6.0 & 6.0 \\ 3.0 & 7.0 & 3.0 & 7.0 \\ 2.0 & 9.0 & 2.0 & 9.0 \\ 5.0 & 5.0 & 3.0 & 3.0 \end{bmatrix}$ num. variables: 4 range: $[0;10]$ fmin: ≈ -10.40	

(Continued)

Table 2 (continued)

Benchmark Function	3D Plot
F23–Shekel-10 Function $f(\mathbf{x}) = -\sum_{i=1}^{10} \frac{1}{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i}$ $\mathbf{c} = [0.1,\ 0.2,\ 0.2,\ 0.4,\ 0.4,\ 0.6,\ 0.3,\ 0.7,\ 0.5,\ 0.5]$ $\mathbf{A} = \begin{bmatrix} 4.0 & 4.0 & 4.0 & 4.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 8.0 & 8.0 & 8.0 & 8.0 \\ 6.0 & 6.0 & 6.0 & 6.0 \\ 3.0 & 7.0 & 3.0 & 7.0 \\ 2.0 & 9.0 & 2.0 & 9.0 \\ 5.0 & 5.0 & 3.0 & 3.0 \\ 8.0 & 1.0 & 8.0 & 1.0 \\ 6.0 & 2.0 & 6.0 & 2.0 \\ 7.0 & 3.6 & 7.0 & 3.6 \end{bmatrix}$ num. variables: 4 range: $[0;10]$ fmin: ≈ -10.53	

Tables 3 and 4 report the comparative performance of SRCA against well-established metaheuristic algorithms, highlighting its relative effectiveness on benchmark functions. In the tables, the symbol [*] refers to results obtained from Python implementations specifically developed as part of this study. Conversely, the symbol [**] denotes results sourced directly from established literature [9]. Relying on peer-reviewed baseline data for canonical algorithms (e.g., PSO, WOA, GSA) strictly prevents potential "implementation bias", ensuring these comparators are represented at their objectively verified best under the specified computational constraints. In contrast, the GSO and FA implementations required local execution because they incorporate specific recent extensions [22] for which comprehensive baseline data across all 23 functions was unavailable. In particular, The GSO algorithm was extended beyond its standard formulation through several methodological enhancements [22]. Four distinct exploration strategies were introduced for the rangers (global, local, cycle-by-cycle hybrid, and periodic switching) enabling a dynamic balance between exploration and exploitation. Additionally, producers execute directional searches via adaptive perturbations in spherical coordinates, with a decaying amplitude mechanism that encourages convergence while mitigating premature stagnation. A directional reset mechanism embedded within the producer simulates scouting behavior, further improving adaptability during extended optimization processes. The GSO algorithm was configured with 50 producers, each associated with 20 scroungers and 50 rangers, resulting in a structured and hierarchical search process that balances directed exploitation with distributed exploration. A customized Firefly Algorithm (FA) was also implemented in Python to reflect the exploration logic adopted in SRCA and the revised GSO. In addition to the canonical attractiveness-and-randomness scheme, the proposed FA integrates four interchangeable random exploration strategies controlled by the same parameters used in GSO. At each iteration, every firefly is also attracted toward the centroid of the x_{elite} brightest individuals (x_{elite} = 5 by default), yielding a multi-leader attraction mechanism similar to the ensemble option in SRCA. The base attractiveness β_0, light absorption coefficient γ, random step amplitude

α, and local search radius all decay linearly over time, promoting broad exploration in early stages and fine-grained exploitation in later iterations. An elitist replacement step ensures that the current global best (queen firefly) is never lost, while on-the-fly random scouting injects diversity and mitigates stagnation. The FA was configured with an initial population of 100 fireflies, complemented by 100 additional random solutions evaluated at each iteration under various random movement strategies.

Table 3: Benchmark results for GSO, FA, and PSO algorithms.

	SRCA			GSO*		FA*		PSO**	
	Min	**Ave**	**Std**	**Ave**	**Std**	**Ave**	**Std**	**Ave**	**Std**
F1	1.28E-04	1.63E-04	1.40E-05	1.07E-02	5.26E-03	7.50E-02	7.94E-03	1.36E-04	2.02E-04
F2	0.0505	0.0552	0.0027	0.0206	0.0058	0.1770	0.0025	0.0421	0.0454
F3	0.789	7.620	2.526	16.420	7.488	4.371	1.215	70.126	22.119
F4	0.0060	0.2309	0.3141	2.0427	0.3520	0.4317	0.1899	1.0865	0.3170
F5	5.996	25.181	5.040	74.245	52.837	65.226	48.417	96.718	60.116
F6	0	0	0	0	0	2.833	0.753	1.02E-04	8.28E-05
F7	0.0025	0.0102	0.0037	2.59E-04	9.41E-05	0.0070	0.0027	0.1229	0.0450
F8	−10791.6	−9698.5	423.4	−7515.3	272.1	−8345.7	626.5	−4841.3	1152.8
F9	41.818	63.511	9.634	50.399	2.861	122.188	34.387	46.704	11.629
F10	0.0086	0.0098	0.0005	0.1019	0.0291	0.4072	0.6362	0.2760	0.5090
F11	9.96E-06	0.0128	0.0149	0.0698	0.0212	0.2864	0.0402	0.0092	0.0077
F12	2.0E-06	0.8421	1.0205	0.0002	6.34E-05	0.7463	0.9394	0.0069	0.0263
F13	2.81E-05	3.51E-05	4.0E-06	0.0041	0.0029	0.0120	0.0062	0.0067	0.0089
F14	0.998	0.998	9.95E-16	0.998	0	0.998	6.66E-12	3.627	2.561
F15	0.0005	0.0008	0.0002	0.0044	5.98E-09	0.0044	3.36E-07	N/A	N/A
F16	−1.0316	−1.0316	9.44E-10	−1.0316	0	−1.0316	6.36E-08	−1.0316	6.25E-16
F17	0.3979	0.3979	5.35E-10	0.3979	0	0.3979	6.16E-08	0.3979	0
F18	3.000	3.000	7.57E-08	3.000	0	3.000	1.15E-06	3.000	1.33E-15
F19	−3.8628	−3.8628	4.14E-07	−3.8628	0	−3.8628	1.36E-08	−3.8628	2.58E-15
F20	−3.3224	−3.2093	0.0242	−3.3224	1.35E-08	−3.2985	0.0533	−3.2663	0.0605
F21	−10.1532	−10.1532	2.03E-06	−10.1532	0	−10.1532	6.91E-05	−6.8651	3.0196
F22	−10.4029	−10.4029	3.59E-06	−10.4029	1.0E-06	−10.4029	8.11E-05	−8.4565	3.0871
F23	−10.5364	−10.5363	0.0006	−10.5364	0	−10.5364	2.20E-05	−9.9529	1.7828

Table 4: Benchmark results for WOA, GSA, DE, and FEP algorithms.

	WOA**		GSA**		DE**		FEP**	
	Ave	**Std**	**Ave**	**Std**	**Ave**	**Std**	**Ave**	**Std**
F1	1.41E-30	4.91E-30	2.53E-16	9.67E-17	8.2E-14	5.9E-14	0.00057	0.00013
F2	1.06E-21	2.39E-21	0.0557	0.1941	1.5E-09	9.9E-10	0.0081	0.00077
F3	5.39E-07	2.93E-06	896.535	318.956	6.8E-11	7.4E-11	0.016	0.014
F4	0.0726	0.3975	7.3549	1.7415	0	0	0.3	0.5
F5	27.866	0.7636	67.543	62.225	0	0	5.06	5.87
F6	3.116	0.532	0	1.74E-16	0	0	0	0
F7	0.0014	0.0011	0.0894	0.0434	0.0046	0.0012	0.1415	0.3522
F8	−5080.8	695.8	−2821.1	493.0	−11080.1	574.7	−12554.5	52.6
F9	0	0	25.968	7.470	69.2	38.8	0.046	0.012
F10	7.404	9.898	0.0621	0.2363	0	0	0.018	0.0021

(Continued)

Table 4 (continued)

	WOA**		GSA**		DE**		FEP**	
	Ave	Std	Ave	Std	Ave	Std	Ave	Std
F11	0.0003	0.0016	27.702	5.040	0	0	0.016	0.022
F12	0.3397	0.2149	1.7996	0.9511	7.9E-15	8.0E-15	9.2E-06	3.6E-06
F13	1.8890	0.2661	8.8991	7.1262	5.1E-14	4.8E-14	0.00016	7.3E-05
F14	2.1120	2.4986	5.8598	3.8313	0.9980	3.3E-16	1.22	0.56
F15	N/A	N/A	0.0037	0.0016	N/A	0.0003	N/A	N/A
F16	−1.0316	4.2E-07	−1.0316	4.88E-16	−1.0316	3.1E-13	−1.03	4.9E-07
F17	0.3979	2.7E-05	0.3979	0	0.3979	9.9E-09	0.398	1.5E-07
F18	3.000	4.22E-15	3.000	4.17E-15	3.000	2.0E-15	3.02	0.11
F19	−3.8562	0.0027	−3.8628	2.29E-15	N/A	N/A	−3.86	1.4E-05
F20	−2.9811	0.3767	−3.3178	0.0231	N/A	N/A	−3.27	0.059
F21	−7.0492	3.6296	−5.9551	3.7371	−10.1532	2.5E-06	−5.52	1.59
F22	−8.1818	3.8292	−9.6845	2.0141	−10.4029	3.9E-07	−5.53	2.12
F23	−9.3424	2.4147	−10.5364	2.6E-15	−10.5364	1.9E-07	−6.57	3.14

Analyzing the results presented in Tables 3 and 4, it can be observed that SRCA consistently approaches the global optimum across a wide variety of search spaces. It is important to objectively note that SRCA does not universally dominate the benchmarks. On certain unimodal and continuous functions (such as F1, F2, F3, and F9), highly specialized algorithms like DE and WOA exhibit superior precision, reaching extreme minima or exact zeroes. Similarly, GSO and PSO outperform SRCA on specific multimodal topologies (e.g., F7, F11, F20). However, SRCA achieves the best results among the compared methods on 14 out of the 23 functions, showcasing a highly balanced exploitation capacity and robust convergence behavior. The deterministic movement module, when combined with the stochastic cycle-based exploration and the adaptive strategy selection, creates a powerful synergy that prevents stagnation and fosters reliable general-purpose optimization.

However, it is important to rigorously analyze the specific behavioral dynamics observed on function F12, a complex benchmark characterized by sharp boundary effects and severe penalty terms. The structural difficulty SRCA occasionally encounters here stems directly from its architectural mechanics. Specifically, the heavy-tailed stochastic operators (such as the Lévy flight) frequently generate large exploratory steps. In a landscape bounded by steep penalty walls, these massive jumps cause frequent boundary violations. While algorithms like PSO naturally mitigate this issue through velocity clamping (which smoothly decelerates particles near the boundaries), SRCA's current boundary-handling mechanism simply truncates variables at the bounds (clipping). This approach can temporarily cause an artificial concentration of solutions on the boundary, resulting in localized losses of population diversity. However, it must be noted that when SRCA is evaluated under a strictly normalized and equitable computational budget (MaxFEs), its adaptive strategies are granted sufficient evaluations to recover from these boundary collisions, yielding significantly improved final results. As seen in Table 3, while SRCA does not surpass the extreme precision of GSO or PSO on this specific function, adopting the proper MaxFEs budget substantially narrowed the performance gap, validating the algorithm's resilience on penalized spaces. Nevertheless, this analysis highlights that the integration of advanced boundary-handling or gradient-aware repair mechanisms remains a valuable future improvement for navigating sharp penalty boundaries.

While this isolated issue is observed, SRCA demonstrates a competitive trade-off between convergence speed and solution quality, with low standard deviation values across most runs, indicating high robustness and reliability.

4.2 Performance Evaluation on Benchmark Structural Optimization Problems

To rigorously assess the practical applicability and constraint-handling capabilities of the proposed SRCA framework, a suite of six classic structural engineering design problems was selected [29–31]. These problems are widely recognized in the literature for their highly non-convex search spaces, non-linear constraints, and mixed continuous-discrete decision variables, making them ideal stress tests for modern metaheuristics. The evaluated problems are: Tension/Compression Spring Design (F1), Pressure Vessel Design (F2), Three-Bar Truss Design (F3), Welded Beam Design (F4), Multiple Disk Clutch Brake Design (F5), and Speed Reducer Design (F6). The detailed mathematical formulations, including the objective functions, constraint inequalities, and decision variable bounds for all six optimization problems, are thoroughly documented in existing literature [32,33].

In this experimental phase, SRCA was comprehensively compared against an extensive set of advanced algorithms from the state-of-the-art, including SASS, COLSHADE, IUDE, BiPopEpsMAgES, ϵMAgES, iLSHADE, SRA, GA3, GA4, HPSO, CPSO, and MBA. The performance metrics evaluated across multiple independent runs include the Best, Mean, Worst, and Standard Deviation (Std_Dev) of the objective function evaluations. The consolidated numerical results are presented in Table 5.

Table 5: Comprehensive comparison of optimization results for structural design problems (F1-F6).

Method	Stat	F1 *Spring*	F2 *Pressure*	F3 *Truss*	F4 *Welded*	F5 *Clutch*	F6 *Reducer*
SRCA	Best	0.012734	6070.58	263.896	1.72852	0.235243	2996.61
	Mean	0.012836	6279.28	263.897	1.73914	0.235243	2997.04
	Std_Dev	$4.50 \cdot 10^{-5}$	143.164	0.000931	0.015855	$1.31 \cdot 10^{-7}$	0.452634
	Worst	0.012928	6366.45	263.897	1.79826	0.235243	2998.38
SASS	Best	0.0127	6060	263.896	1.67	0.235242	2990
	Mean	0.0127	6410	263.896	1.68	0.235242	3000
	Std_Dev	$2.62 \cdot 10^{-6}$	628	$5.69 \cdot 10^{-14}$	0.0201	$8.55 \cdot 10^{-7}$	6.29
	Worst	0.0127	8960	263.896	1.79	0.235242	3050
COLSHADE	Best	0.0127	6060	263.896	1.67	0.235242	2990
	Mean	0.0127	6060	263.896	1.67	0.235242	2990
	Std_Dev	$1.08 \cdot 10^{-7}$	8.53	$5.80 \cdot 10^{-14}$	$2.27 \cdot 10^{-16}$	$2.83 \cdot 10^{-17}$	$4.64 \cdot 10^{13}$
	Worst	0.0127	6090	263.896	1.67	0.235242	2990
IUDE	Best	0.0127	6060	264	1.67	0.235242	2990
	Mean	0.0127	6060	264	1.67	0.235242	2990
	Std_Dev	$1.08 \cdot 10^{-5}$	6.16	0	$1.20 \cdot 10^{-16}$	$1.69 \cdot 10^{-16}$	$4.64 \cdot 10^{13}$
	Worst	0.0127	6060	264	1.67	0.235242	2990
BiPopEpsMAgES	Best	0.0127	6060	263.896	1.67	0.235242	2990
	Mean	0.0127	6170	263.896	1.67	0.235242	2990
	Std_Dev	$1.09 \cdot 10^{-4}$	210	$8.71 \cdot 10^{-5}$	0.0023	$5.84 \cdot 10^{-16}$	$4.64 \cdot 10^{13}$
	Worst	0.0137	7460	263.896	1.67	0.235242	2990
ϵMAgES	Best	0.0127	6060	264	1.67	0.235242	2990
	Mean	0.0127	7380	265	1.67	0.235242	2990
	Std_Dev	$2.16 \cdot 10^{-4}$	1930	2.88	0.0395	$1.69 \cdot 10^{-16}$	$4.64 \cdot 10^{13}$

(Continued)

Table 5 (continued)

Method	Stat	F1 *Spring*	F2 *Pressure*	F3 *Truss*	F4 *Welded*	F5 *Clutch*	F6 *Reducer*
	Worst	0.0137	11900	264	1.85	0.235242	2990
iLSHADE	Best	0.0127	6060	264	1.67	0.235242	2990
	Mean	0.013	8480	264	1.67	0.235242	2990
	Std_Dev	0.00106	3140	0.0199	$7.59 \cdot 10^{-7}$	$1.13 \cdot 10^{-16}$	$4.64 \cdot 10^{13}$
	Worst	0.0178	14900	264	1.67	0.235242	2990
SRA	Best	0.012667	6059.72	263.463	1.6694	0.235242	2995
	Mean	0.013706	6708.43	263.463	1.745	0.246744	2997
	Std_Dev	0.01467	509.304	$8.21 \cdot 10^{-9}$	0.24143	0.0090697	1.951
	Worst	0.017414	7544.49	263.463	1.926	0.262215	3002
GA3	Best	0.0127048	6288.74	—	—	—	—
	Mean	0.012769	6293.84	—	—	—	—
	Std_Dev	$3.94 \cdot 10^{-5}$	7.4133	—	—	—	—
	Worst	0.012822	6308.5	—	—	—	—
GA4	Best	0.012681	6059.95	—	—	—	—
	Mean	0.012742	6177.25	—	—	—	—
	Std_Dev	$5.90 \cdot 10^{-5}$	130.93	—	—	—	—
	Worst	0.012973	6469.32	—	—	—	—
HPSO	Best	0.0126652	6059.71	—	—	—	—
	Mean	0.0127072	6099.93	—	—	—	—
	Std_Dev	$1.58 \cdot 10^{-5}$	86.2	—	—	—	—
	Worst	0.012719	6288.68	—	—	—	—
CPSO	Best	0.0126747	6061.08	—	—	—	—
	Mean	0.01273	6147.13	—	—	—	—
	Std_Dev	$5.20 \cdot 10^{-4}$	86.45	—	—	—	—
	Worst	0.012924	6363.8	—	—	—	—
MBA	Best	0.012665	5889.32	—	—	—	2994.48
	Mean	0.012713	6200.65	—	—	—	2996.77
	Std_Dev	$6.30 \cdot 10^{-5}$	160.34	—	—	—	1.56
	Worst	0.0129	6392.51	—	—	—	2999.65

An analysis of the results presented in Table 5 reveals that SRCA is a reliable and competitive framework for real-world engineering design. While highly specialized algorithms occasionally report marginally lower extreme minima, SRCA distinguishes itself through operational stability and robustness across independent evaluations, which are critical features for industrial applications where reproducibility is paramount.

For the Tension/Compression Spring Design (F1), SRCA achieves a best cost of 0.012734, grouping tightly with top-tier methods like COLSHADE and IUDE (0.01270). Notably, SRCA maintains a low standard deviation ($4.50 \cdot 10^{-5}$), confirming that its adaptive stochastic operators effectively explore the narrow feasible region of this specific problem without undergoing divergent behaviors.

In the Pressure Vessel Design (F2), the algorithm locates a highly competitive minimum of 6070.58, approaching the prevalent optimum cluster established around 6060. While top-tier variants like COLSHADE and IUDE achieve a superior mean cost (6060), SRCA's operational consistency remains noteworthy: its mean cost of 6279.28 outperforms several complex competitors, such as SASS (6410), ϵMAgES (7380), and iLSHADE (8480). These latter algorithms suffer from severe standard deviations (up to 3140 for iLSHADE),

indicating occasional convergence failures. Conversely, SRCA systematically secures high-quality feasible solutions without such extreme variance.

The Three-Bar Truss Design (F3) highlights SRCA's precision. The algorithm perfectly matches the optimal global value of 263.896 discovered by top-performing approaches like COLSHADE and BiPopEps-MAgES. The near-zero standard deviation (0.000931) underlines its capacity to manage non-linear inequality constraints seamlessly.

Regarding the Welded Beam Design (F4), SRCA records a best cost of 1.72852. While slightly higher than the 1.67 optimum found by differential evolution variants, the tight gap between its Best (1.728) and Worst (1.798) runs proves the absence of stagnation in sub-optimal local traps, a common issue for this highly multimodal benchmark.

For the Multiple Disk Clutch Brake (F5), SRCA's performance is virtually indistinguishable from the known global optimum. It identifies a minimum of 0.235243 (compared to 0.235242), achieving an exceptional standard deviation of $1.31 \cdot 10^{-7}$. This demonstrates that the algorithm's continuous boundary handling and dynamic parameter scaling are calibrated for discrete/continuous mixed spaces.

Finally, in the Speed Reducer Design (F6), SRCA achieves a best result of 2996.61. While methods like SRA (2995) and MBA (2994.48) locate marginally lower extreme minima, a notable aspect of SRCA's performance in this benchmark is the consistency across independent runs, evidenced by a standard deviation of 0.4526. In comparison, several competing methods (such as COLSHADE, IUDE, and BiPopEpsMAgES) record significantly higher variance levels (Std_Dev $\approx 10^{13}$) for this specific problem, which may indicate sensitivity to initialization or difficulties in managing the problem's constraints. The numerical results suggest that SRCA's dynamic scaling of stochastic steps helps maintain search stability, preventing divergent behaviors and facilitating reliable convergence within the feasible domain.

To exhaustively address the verification of structural constraints and provide complete reproducibility, Table 6 explicitly reports the optimal decision vectors (design variables) corresponding to the best solutions achieved by SRCA for each evaluated problem. Furthermore, the table formally confirms the feasibility status, verifying that all reported optimal solutions strictly satisfy the respective non-linear mechanical and geometric inequality constraints without any violations.

Table 6: Optimal design variables and feasibility status for the structural engineering problems obtained by SRCA.

Problem	Optimal Decision Vector x	Best Cost	Feasible
F1-Tension/Compression Spring	[0.0500, 0.3173, 14.0518]	0.012734	Yes
F2-Pressure Vessel	[0.8125, 0.4375, 42.0097, 177.7408]	6070.58	Yes
F3-Three-Bar Truss	[0.7883, 0.4092]	263.896	Yes
F4-Welded Beam	[0.2038, 3.5129, 9.0376, 0.2058]	1.72852	Yes
F5-Multiple Disk Clutch Brake	[70.0000, 90.0000, 1.0000, 829.8915, 2]	0.235243	Yes
F6-Speed Reducer	[3.5002, 0.7000, 17, 7.3010, 7.8000, 3.3509, 5.2867]	2996.61	Yes

In conclusion, the structural benchmark analysis confirms that the proposed integration of directional deterministic exploitation and dynamically scheduled stochastic exploration prevents premature convergence and ensures stable constraint satisfaction across varied engineering topologies.

4.3 Ablation Study

To justify the architectural complexity of SRCA and scientifically validate the inclusion of its distinct movement and exploration strategies, an ablation study was conducted. Following standard experimental practices, three representative benchmark functions were selected: F1 (unimodal, testing pure convergence speed), F10 (multimodal, testing local optima avoidance), and F22 (highly asymmetrical local traps).

The fully equipped SRCA ("Full") was compared against three structurally disabled variants: SRCA "w/o Spiral" (disabling the spiral directional movement, $\omega_2 = 0$), SRCA "w/o Adapt" (disabling the adaptive data-driven movement, $\omega_3 = 0$), and SRCA "w/o Lévy" (preventing the adaptive controller from utilizing heavy-tailed stochastic jumps). All variants were executed for 25 independent runs under the strict MaxFEs budget normalization detailed in Section 4. The results are consolidated in Table 7.

Table 7: Ablation study results (Mean and Standard Deviation) for F1, F10, and F22 benchmarks.

Variant	F1 (Unimodal)		F10 (Multimodal)		F22 (Asymmetrical)	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
SRCA (Full)	1.63E-04	1.39E-05	9.76E-03	5.36E-04	−10.4029	3.59E-06
SRCA w/o Spiral	1.19E-01	3.03E-01	7.02E-01	6.60E-01	−9.6300	1.9952
SRCA w/o Adapt	2.49E-04	4.29E-05	1.18E-02	1.15E-03	−9.2886	2.2423
SRCA w/o Lévy	1.91E-04	1.88E-05	6.94E-02	2.92E-01	−8.4960	2.5426

The data demonstrates that every architectural component contributes significantly to the framework's overall robustness. Disabling the spiral movement (w/o Spiral) causes severe performance degradation and extreme variance across all landscape types, proving its necessity for localized spatial navigation. Disabling the heavy-tailed stochastic jumps (w/o Lévy) crucially impairs the algorithm's ability to escape local minima, as evidenced by the dramatic loss of precision and stability on the multimodal F10 and F22 benchmarks. Finally, disabling the adaptive movement (w/o Adapt) leads to a consistent loss of precision and sharply increased standard deviations, particularly on asymmetrical landscapes. Ultimately, this ablation study empirically demonstrates that the optimal performance of SRCA relies on the synergistic integration of all its structural components.

5 Conclusions

This study introduced the Structured Random Cycle-guided Algorithm (SRCA), an architectural synthesis that acts as a unified adaptive framework for operator scheduling, combining structured directionally-guided movements with a diverse ensemble of randomized exploration strategies. By embedding these components in a cycle-based framework and enabling adaptive control based on population feedback, SRCA is designed to maintain a robust balance between global exploration and local exploitation across a wide range of optimization search spaces. Quantitative validation across 23 standard benchmark functions and six highly constrained engineering design problems was conducted under strictly identical experimental conditions (normalized by maximum function evaluations, MaxFEs) against state-of-the-art competitors, demonstrating the algorithm's adaptability and potential for general-purpose optimization.

On benchmark functions, SRCA achieved best-known or near-optimal solutions in 18 out of 23 cases, demonstrating consistent robustness across both unimodal and multimodal landscapes. It showed competitive performance compared to FA, PSO, WOA, and GSA, particularly in terms of average results and convergence stability. On challenging multimodal problems such as F3, F4, and F9, SRCA exhibited strong

global search capabilities with low variance across runs. For high-dimensional or hybrid functions (e.g., F11, F13, F20-F23), it maintained a high level of precision and solution consistency, reflecting an effective balance between diversification and intensification achieved through its hybrid search framework.

The algorithm was also validated on six classical engineering design problems (including Tension/Compression Spring, Speed Reducer, and Multiple Disk Clutch Brake). SRCA successfully reached or perfectly matched known global optima in multiple cases, with exceptionally stable convergence and effective constraint satisfaction. Compared to advanced state-of-the-art metaheuristics:

- It matched or outperformed highly specialized algorithms (such as COLSHADE, IUDE, and BiPopEps-MAgES) in solution quality and precision;
- It displayed significantly greater operational robustness and lower variance (standard deviation) than several modern variants (e.g., iLSHADE, SASS), particularly in discrete or tightly constrained scenarios;
- It retained broad, general-purpose applicability without requiring problem-specific constraint-handling heuristics;
- It complemented classical algorithms like GSO and PSO by offering faster early-stage convergence and improved resilience against stagnation.

Additionally, a dedicated ablation study validated the algorithmic architecture, proving that the synergistic integration of spiral navigation, adaptive data-driven movements, and heavy-tailed Lévy flights is justify to prevent premature stagnation and ensure robust performance across diverse topological landscapes.

While boundary handling on landscapes with severe constraint penalties (e.g., F12) initially posed a structural challenge due to strict variable clipping, evaluation under equitable computational budgets (MaxFEs) confirmed the framework's adaptive capacity to recover and yield highly competitive results. Nevertheless, future work will focus on integrating advanced boundary-handling and gradient-aware repair mechanisms to further streamline navigation in highly penalized regions. Furthermore, in terms of practical applicability, the adaptive nature of SRCA makes it particularly suitable for real-world engineering scenarios beyond the tested benchmarks. Its ability to dynamically balance exploration and exploitation without extensive parameter tuning suggests potential applications in the optimal sizing of mechanical components, the tuning of industrial control systems, and the design of smart structural devices, where objective functions are often non-differentiable or computationally expensive black-box models.

Overall, SRCA emerges as a flexible and effective optimizer for both continuous and discrete domains, capable of addressing complex, real-world problems without the need for domain-specific heuristics. Rather than proposing a single new search operator, SRCA demonstrates how structured cycling and adaptive operator selection can be combined into a coherent optimization paradigm, offering a flexible foundation for future intelligent optimization frameworks. Further developments will focus on scaling to large-dimensional scenarios and refining the adaptive controller to enhance generalization.

Acknowledgement: The authors have no specific acknowledgements to declare.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm their contribution to the paper as follows. Conceptualization, methodology design, investigation, and original draft writing: Giuseppe Marannano and Tommaso Ingrassia; software development, methodology validation, and visualization: Giuseppe Marannano and Antonino Cirello. All authors actively participated in managing and interpreting the results, manuscript revision, and technical discussions. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The numerical data supporting the findings of this study are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable. This study does not involve human or animal subjects.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sarker RA, Newton CS. Optimization modelling: a practical approach. Boca Raton, FL, USA: CRC Press; 2007. doi:10.1201/9781420043112.
2. Kaveh A, Ghazaan MI. Meta-heuristic algorithms for optimal design of real-size structures. Cham, Switzerland: Springer; 2018. doi:10.1007/978-3-319-78780-0.
3. Kumar A, Nadeem M, Banka H. Nature inspired optimization algorithms: a comprehensive overview. Evol Syst. 2023;14(1):1–31. doi:10.1007/s12530-022-09432-6.
4. Yang X-S, Deb S. Engineering optimisation by cuckoo search. Int J Math Model Numer Optim. 2010;1(4):330–43. doi:10.1504/ijmmno.2010.035430.
5. He S, Wu QH, Saunders JR. A novel group search optimizer inspired by animal behavioural ecology. In: 2006 IEEE International Conference on Evolutionary Computation; 2006 Jul 16–21; Vancouver, BC, Canada. p. 1272–8. doi:10.1109/CEC.2006.1688455.
6. Li L, Liu F. Optimum design of structures with group search optimizer algorithm. In: Group search optimization for applications in structural design. Adaptation, Learning, and Optimization. Vol. 9. Berlin, Heidelberg, Germany: Springer; 2011. p. 123–64. doi:10.1007/978-3-642-20536-1_4.
7. Marannano G, Ingrassia T, Ricotta V, Nigrelli V. Numerical optimization of a composite sandwich panel with a novel bi-directional corrugated core using an animal-inspired optimization algorithm. In: Lecture notes in mechanical engineering. Cham, Switzerland: Springer; 2023. p. 565–75. doi:10.1007/978-3-031-15928-2_56.
8. Li X, Tian D. The importance of the whale optimization algorithm in addressing cloud computing issues. J Eng Appl Sci. 2025;72(1):162. doi:10.1186/s44147-025-00740-7.
9. Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Softw. 2016;95:51–67. doi:10.1016/j.advengsoft.2016.01.008.
10. Mahmood S, Bawany NZ, Tanweer MR. A comprehensive survey of whale optimization algorithm: modifications and classification. Indones J Electr Eng Comput Sci. 2023;29(2):899–910. doi:10.11591/ijeecs.v29.i2.pp899-910.
11. Rahimnejad A, Akbari E, Mirjalili S, Gadsden SA, Trojovský P, Trojovská E. An improved hybrid whale optimization algorithm for global optimization and engineering design problems. PeerJ Comput Sci. 2023;9:e1557. doi:10.7717/peerj-cs.1557.
12. Rao SS. Engineering optimization: theory and practice. 4th ed. Hoboken, NJ, USA: Wiley; 2009. doi:10.1002/9780470549124.
13. Makhadmeh SN, Awadallah MA, Kassaymeh S, Al-Betar MA, Sanjalawe Y, Kouka S, et al. Recent advances in multi-objective cuckoo search algorithm, its variants and applications. Arch Comput Methods Eng. 2025;32(5):3213–40. doi:10.1007/s11831-025-10240-9.
14. Shlesinger MF, Zaslavsky GM, Klafter J. Strange kinetics. Nature. 1993;363(6424):31–7. doi:10.1038/363031a0.
15. Ma L, Dai C, Xue X, Peng C. A multi-objective particle swarm optimization algorithm based on decomposition and multi-selection strategy. Comput Mater Contin. 2025;82(1):997–1026. doi:10.32604/cmc.2024.057168.
16. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks. Piscataway, NJ, USA: IEEE; 1995. Vol. 4, p. 1942–8. doi:10.1109/ICNN.1995.488968.
17. Wang D, Tan D, Liu L. Particle swarm optimization algorithm: an overview. Soft Comput. 2018;22(2):387–408. doi:10.1007/s00500-016-2474-6.
18. Yang XS. Firefly algorithm, stochastic test functions and design optimization. Int J Bio-Inspired Comput. 2010;2(2):78–84. doi:10.1504/IJBIC.2010.032124.

19. Fister I, Yang XS, Brest J. A comprehensive review of firefly algorithms. Swarm Evol Comput. 2013;13(3):34–46. doi:10.1016/j.swevo.2013.06.001.
20. Zare M, Ghasemi M, Zahedi A, Golalipour K, Mohammadi SK, Mirjalili S, et al. A global best-guided firefly algorithm for engineering problems. J Bionic Eng. 2023;20:1102–15. doi:10.1007/s42235-023-00386-2.
21. Scalia GLa, Micale R, Giallanza A, Marannano G. Firefly algorithm based upon slicing structure encoding for unequal facility layout problem. Int J Ind Eng Comput. 2019;10:215–26. doi:10.5267/j.ijiec.2019.2.003.
22. Cirello A, Ingrassia T, Mancuso A, Marannano G, Mirulla AI, Ricotta V. Evaluating the efficiency of nature-inspired algorithms for finite element optimization in the ANSYS environment. Appl Sci. 2025;15(12):6750. doi:10.3390/app15126750.
23. Wei Y, Song P, Luo Q, Zhou Y. Differential evolution with improved equilibrium optimizer for combined heat and power economic dispatch problem. Comput, Mat Cont. 2025;85(1):1235–65. doi:10.32604/cmc.2025.066527.
24. Zhan Z-H, Zhang J, Li Y, Chung HS-H. Adaptive particle swarm optimization. IEEE Trans Syst Man, Cybernet, Part B (Cybernetics). 2009;39(6):1362–81. doi:10.1109/TSMCB.2009.2015956.
25. Hop DC, Hop NVan, Anh TTM. Adaptive particle swarm optimization for integrated quay crane and yard truck scheduling problem. Comput Indust Eng. 2021;153:107075. doi:10.1016/j.cie.2020.107075.
26. Mantegna RN. Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. Phys Rev E. 1994;49(5):4677–83. doi:10.1103/PhysRevE.49.4677.
27. Mohammed HM, Umar SU, Rashid TA. A systematic and meta-analysis survey of whale optimization algorithm. Comput Intell Neurosci. 2019. doi:10.1155/2019/8718571.
28. Tang J, Wang L. A whale optimization algorithm based on atom-like structure differential evolution for solving engineering design problems. Sci Rep. 2024;14(1):1730. doi:10.1038/s41598-023-51135-8.
29. Yang XS, Huyck C, Karamanoglu M, Khan N. True global optimality of the pressure vessel design problem: a benchmark for bio-inspired optimization algorithms. Int J Bio-Inspired Comput. 2013;5(6):329–35. doi:10.1504/ijbic.2013.058910.
30. Sandgren E. Nonlinear integer and discrete programming in mechanical design optimization. J Mech Des. 1990;112(2):223–9. doi:10.1115/1.2912596.
31. Cagnina LC, Esquivel SC, Coello Coello CAC. Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica. 2008;32(3):319–26. doi:10.1080/03052150802265870.
32. Hussein NK, Qaraad M, Najjar AMEl, Farag MA, Elhosseini MA, Mirjalili S, et al. Schrödinger optimizer: a quantum duality-driven metaheuristic for stochastic optimization and engineering challenges. Knowl-Based Syst. 2025;328:114273. doi:10.1016/j.knosys.2025.114273.
33. El-Shorbagy MA, Elazeem AMA. Convex combination search algorithm: a novel metaheuristic optimization algorithm for solving global optimization and engineering design problems. J Eng Res (Kuwait). 2025;13(3):2320–39. doi:10.1016/j.jer.2024.05.008.