

Adaptive Fusion Optimization (AFO): A Hyper-Adaptive Metaheuristic with Learning-Driven Control for Global Minimization

Optimización de Fusión Adaptativa (AFO): una metaheurística hiperadaptativa con control basado en aprendizaje para la minimización global

Manisha Prasad¹ ✉, Amir Khusru Akhtar¹ ✉

¹Faculty of Computing and IT, Usha Martin University. Ranchi, Jharkhand, India – 835103.

Cite as

Prasad M, Khusru Akhtar A. Adaptive Fusion Optimization (AFO): A Hyper-Adaptive Metaheuristic with Learning-Driven Control for Global Minimization. SAP Gamification and Augmented Reality. 2026; 4:276. <https://doi.org/10.62486/gr2026276>

Publication History

Submitted: 28-11-2025

Revised: 17-01-2026

Accepted: 21-03-2026

Published: 22-03-2026

Corresponding Author

Manisha Prasad ✉

ABSTRACT

Introduction: global minimization in complex, nonlinear search spaces is challenging due to premature convergence and high parameter sensitivity in classical metaheuristics.

Objective: this study proposes Adaptive Fusion Optimization (AFO), a hyper-adaptive metaheuristic designed to balance exploration and exploitation through dynamic operator control and online feedback.

Method: AFO integrates a compact fusion pool of three operators with a self-evolving exploration factor and a lightweight learning-driven controller for operator selection. The algorithm adapts its search behavior using fitness improvement and population diversity, avoiding fixed switching rules. Performance is evaluated on standard benchmark functions under a fixed function-evaluation budget and compared with GA, PSO, DE, and ACO.

Results: AFO demonstrates superior accuracy, faster convergence, and stronger robustness, especially on multimodal functions. In 30 dimensions, it achieves mean final fitness of 3.1×10^{-2} on Rastrigin and 6.2×10^{-3} on Ackley. In 50 dimensions, robustness remains high, with Ackley showing a standard deviation of about 4.1×10^{-3} over 30 runs. Statistical tests at 95 % confidence confirm the improvement, with Friedman ranking placing AFO first (mean rank 1.20, $p = 2.1 \times 10^{-6}$), supported by Wilcoxon pairwise tests.

Conclusions: AFO provides a reliable framework for global minimization and shows promise for extension to constrained, multi-objective, and real-world optimization tasks such as energy-efficient scheduling and power dispatch.

Keywords: Adaptive Fusion Optimization; Hyper-Adaptive Metaheuristic; Learning-Driven Operator Selection; Global Minimization; Benchmark Test Functions.

RESUMEN

Introducción: la minimización global en espacios de búsqueda complejos y no lineales presenta desafíos debido a la convergencia prematura y la alta sensibilidad a los parámetros en las metaheurísticas clásicas.

Objetivo: este estudio propone la Optimización por Fusión Adaptativa (AFO), una metaheurística hiperadaptativa diseñada para equilibrar la exploración y la explotación mediante el control dinámico de operadores y la retroalimentación en línea.

Método: AFO integra un conjunto compacto de tres operadores con un factor de exploración autoevolutivo y un controlador ligero basado en aprendizaje para la selección de operadores. El algoritmo adapta su comportamiento de búsqueda mediante la mejora de la aptitud y la diversidad de la población, evitando reglas de conmutación fijas. El rendimiento se evalúa en funciones de referencia estándar con un presupuesto fijo de evaluación de funciones y se compara con GA, PSO, DE y ACO.

Resultados: AFO demuestra una precisión superior, una convergencia más rápida y una mayor robustez, especialmente en funciones multimodales. En 30 dimensiones, alcanza una aptitud final media de 3.1×10^{-2} en Rastrigin y de 6.2×10^{-3} en Ackley. En 50 dimensiones, la robustez se mantiene alta, con Ackley mostrando una desviación estándar de aproximadamente 4.1×10^{-3} en 30 ejecuciones. Las pruebas estadísticas con un nivel de confianza del 95 % confirman la mejora, con la clasificación de Friedman situando a AFO en primer lugar (rango medio 1.20, $p = 2.1 \times 10^{-6}$), respaldada por pruebas de Wilcoxon por pares.

Conclusiones: AFO proporciona un marco fiable para la minimización global y muestra potencial para su extensión a tareas de optimización con restricciones, multiobjetivo y del mundo real, como la programación energéticamente eficiente y la distribución de energía.

Palabras clave: Optimización Adaptativa por Fusión; Metaheurística Hiperadaptativa; Selección de Operadores Basada en Aprendizaje; Minimización Global; Funciones de Prueba de Referencia.

INTRODUCTION

Optimization is a basic but very important problem in science and engineering, where the goal is to find the best solution under given constraints. Many real-world tasks in energy systems, machine learning, networks, routing, and scheduling can be written as global minimization problems.^(1,2) In practice, these problems are often high-dimensional, nonlinear, multimodal, and sometimes noisy, so classical gradient-based methods become less reliable when derivatives are unavailable or unstable.^(3,4)

To handle such difficult landscapes, metaheuristic algorithms are widely used because they are flexible and do not need gradient information. Well-known examples include Genetic Algorithm (GA),⁽⁵⁾ Particle Swarm Optimization (PSO),⁽⁶⁾ Differential Evolution (DE),⁽⁷⁾ and Ant Colony Optimization (ACO).^(8,9) Still, it is well known that no single metaheuristic performs best for all problems, which is consistent with the No Free Lunch idea.⁽¹⁰⁾ A big reason is that many classical metaheuristics depend on fixed control parameters, and these parameters usually need manual tuning. This becomes a problem because early search needs exploration, but later search needs exploitation, and fixed settings often cause premature convergence and stagnation.^(11,12,13)

To improve performance, many researchers proposed hybrid and fusion-based metaheuristics, where multiple operators or algorithms are combined to use complementary strengths.⁽¹⁴⁾ Recent work also reports that hybrid-fusion methods can show strong benchmark performance.⁽¹⁵⁾ But in many cases, switching rules are predefined or combinations are static, which adds complexity and creates extra tuning burden. So robustness becomes an issue.⁽¹⁶⁾

Another strong direction is self-adaptive and learning-driven optimization.^(17,18) Self-adaptive methods adjust parameters during the run using feedback, such as self-adaptive DE variants like jDE and SaDE.⁽¹⁹⁾ These approaches reduce tuning effort, but the adaptation is still mostly rule-based and does not properly “learn” the best behaviour. More recently, reinforcement learning (RL) has been used to guide operator or parameter selection, where the algorithm learns from search feedback. Recent studies indicate that RL-based control can improve convergence and reduce premature convergence,⁽²⁰⁾ but many RL-driven methods become computationally heavy due to complex models, which limits practical scalability.

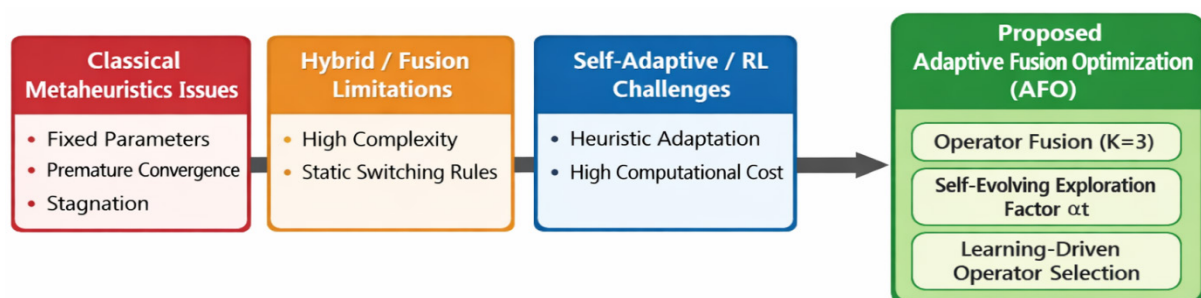


Figure 1. Motivation and gap-to-solution diagram for AFO

From this discussion, the gap is clear: classical metaheuristics lack adaptability, hybrids often become complex, self-adaptive methods are mostly heuristic, and RL-based methods can be costly. So, there is a need of a lightweight but truly adaptive framework that can balance exploration and exploitation dynamically without excessive tuning.

The motivation and research gap leading to the proposed Adaptive Fusion Optimization framework are illustrated in figure 1. This paper proposes Adaptive Fusion Optimization (AFO), a hyper-adaptive metaheuristic for global minimization. AFO uses a compact fusion pool of operators under one framework, a self-evolving exploration factor to control operator behaviour, and a lightweight learning-driven controller to select operators based on improvement and diversity feedback. Unlike fixed hybrids, AFO adapts its strategy during the search instead of following predefined switching rules.

Main contributions of this paper are:

1. A compact hyper-adaptive fusion framework with multiple operators in a unified structure.
2. A self-evolving mechanism that adjusts exploration–exploitation behaviour during the run.
3. A lightweight learning-driven operator selection strategy.
4. Benchmark validation against GA, PSO, DE, and ACO.
5. Statistical validation using Friedman and Wilcoxon tests.

The rest of this paper is organized as follows: Section 2 reviews classical, hybrid, and adaptive metaheuristic algorithms. Section 3 describes the AFO framework. Section 4 presents algorithmic implementation and computational analysis. Section 5 explains the experimental setup. Section 6 reports performance evaluation and statistical analysis. Finally, Section 7 concludes the paper and gives future research directions.

LITERATURE REVIEW

This section summarizes prior work related to the proposed Adaptive Fusion Optimization (AFO). Here classical metaheuristics, hybrid/fusion methods, self-adaptive algorithms, and learning-driven metaheuristics have been covered, and then the gap that motivates AFO has been highlighted.

Classical metaheuristic algorithms such as GA, PSO, DE, and ACO are widely used for global optimization because they are flexible and do not require gradient information.⁽⁹⁾ GA uses evolutionary operators and is generally strong in exploration, but it may converge slowly and lose diversity over time.⁽⁵⁾ PSO converges fast in many cases, but it often suffers from premature convergence in multimodal landscapes.⁽²¹⁾ DE is well-known for continuous optimization performance, but it is sensitive to control parameters like mutation factor and crossover rate.⁽²²⁾ ACO uses pheromone-based learning and works well in many discrete settings, but it can face scalability and stability issues when extended to high-dimensional continuous problems.⁽²³⁾ Overall, the No Free Lunch view also supports that no single method stays best for all problem classes.⁽¹⁰⁾

To improve performance, researchers proposed hybrid and fusion-based metaheuristics, where multiple operators or algorithms are combined to use complementary strengths.⁽¹⁴⁾ Recent work shows hybrid-fusion approaches can outperform single methods on benchmark suites.⁽¹⁶⁾ Still, many hybrids rely on static switching rules or fixed operator weights, which increases complexity and adds tuning burden, so robustness may reduce in new problem settings.⁽²⁴⁾

Another major direction is self-adaptive optimization, where parameters are adjusted automatically using search feedback. Self-adaptive DE variants like jDE and SaDE reduce manual tuning and improve robustness.⁽¹⁹⁾ However, in many cases the update logic remains heuristic and problem-dependent, so the algorithm “reacts” but does not learn the best action properly.⁽²⁵⁾

To make adaptation more intelligent, reinforcement learning (RL) has been integrated into metaheuristics, where an agent selects operators or parameters based on accumulated experience.⁽²⁶⁾ Recent studies report that RL-based control improves convergence and helps in avoiding premature convergence.⁽²⁰⁾ But many RL-driven approaches become computationally heavy due to complex models, and sometimes they overfit to specific tasks.⁽²⁷⁾ A brief comparison of these method families and their limitations is summarized in table 1.

Table 1. Comparison of Metaheuristic Families and Key Limitations

Method Family	Typical Examples	Main Strength	Main Limitation (Gap Point)	Why AFO is needed
Classical metaheuristics	GA, ⁽⁵⁾ PSO, ⁽²¹⁾ DE, ⁽²²⁾ ACO, ⁽²³⁾	Simple, widely applicable	Fixed parameters → stagnation / premature convergence	Need dynamic control
Hybrid / Fusion methods	GA-PSO, ⁽²⁸⁾ memetic ^(14,30) PSO-DE, ⁽²⁹⁾	Better balance in some cases	Static switching, more tuning, complexity	Need adaptive fusion
Self-adaptive methods	jDE, SaDE ⁽¹⁹⁾	Reduces manual tuning	Heuristic updates, not truly learning	Need learning-based selection
Learning-driven methods	RL-controlled optimizers ⁽²⁶⁾	Intelligent adaptation	Often high computation cost / heavy models	Need lightweight learning

So the gap is clear: classical methods lack adaptivity, hybrids become complex, self-adaptive approaches are mostly rule-based, and RL-based methods can be expensive. This directly motivates AFO, which combines compact operator fusion with a self-evolving control factor and a lightweight learning-driven operator selection scheme in one unified framework.

METHOD

Adaptive Fusion Optimization (AFO) Framework

Problem Definition and Notation

Consider a bounded continuous minimization problem:

$$\min_{x \in \Omega} f(x), \Omega = \{x \in \mathbb{R}^D \mid l \leq x \leq u\}$$

Where D is the decision dimension, $l=[l_1, \dots, l_D]$ and $u=[u_1, \dots, u_D]$ are lower and upper bounds, and $f(\cdot)$ is the objective function. Let the population at iteration be:

$$\mathcal{P}^t = \{x_1^t, x_2^t, \dots, x_N^t\}$$

With population size N . The best-so-far solution is:

$$x_*^t = \arg \min_{x \in \mathcal{P}^t} f(x)$$

AFO is designed around three linked ideas:

1. Fusion of multiple search operators (diverse moves in same population),
2. Self-evolving control (parameters change using search feedback),
3. Learning-driven operator selection (a lightweight RL controller decides which operator to apply).

Operator Pool and Fusion Model

AFO maintains a pool of search operators:

$$\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_K\}$$

Each operator maps a parent solution and context to a trial solution:

$$y_i^t = \mathcal{O}_k(x_i^t, \mathcal{P}^t, \theta_k^t, \eta^t)$$

Where:

- θ_k^t are operator-specific parameters (e.g., step size, mutation factor),
- η^t are global adaptive parameters shared across operators.

Fusion in AFO is not a fixed hybrid. It is a dynamic selection among operators where the selection probability changes over time.

Let p_k^t be the probability of choosing operator $\mathcal{O}_k$ at iteration t :

$$p^t = [p_1^t, \dots, p_K^t], \sum_{k=1}^K p_k^t = 1, p_k^t \geq 0$$

For each individual i , AFO selects an operator index:

$$k_i^t \sim \text{Categorical}(p^t)$$

Then generates a trial y_i^t and applies greedy selection:

$$x_i^{t+1} = \begin{cases} y_i^t, & \text{if } f(y_i^t) \leq f(x_i^t) \\ x_i^t, & \text{otherwise} \end{cases}$$

In this paper, we set $K=3$ for a compact fusion pool.

This preserves the best-found quality while allowing exploration through diversified operators.

Representative Operator Templates (Generic Forms; $K=3$ Pool)

AFO uses a compact operator pool $\mathcal{O}=\{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3\}$ to avoid parameter bloat while preserving complementary search behaviors.

Each operator generates a trial vector y_i^t from a parent x_i^t using operator-specific parameters θ_k^t and global adaptive controls η^t .

(a) Differential perturbation operator (DE-like) $\mathcal{O}_1$

Select distinct indices $r_1, r_2, r_3 \in \{1, \dots, N\} \setminus \{i\}$. Mutation:

$$v_i^t = x_{r_1}^t + F^t(x_{r_2}^t - x_{r_3}^t)$$

Binomial crossover (with mandatory index d_{rand}):

$$y_{i,d}^t = \begin{cases} v_{i,d}^t, & \text{if } \text{rand}_d \leq CR^t \text{ or } d = d_{\text{rand}} \\ x_{i,d}^t, & \text{otherwise} \end{cases} \quad d = 1, \dots, D$$

Where $\text{rand}_d \sim U(0,1)$, $F^t \in (F_{\min}, F_{\max})$, and $CR^t \in (CR_{\min}, CR_{\max})$.

(b) Attraction-inertia operator (PSO-like) $\mathcal{O}_2$

AFO maintains velocities $\{v_i^t\}_{i=1}^N$. Update:

$$v_i^{t+1} = \omega^t v_i^t + c_1^t r_1 (pbest_i^t - x_i^t) + c_2^t r_2 (gbest^t - x_i^t)$$

$$y_i^t = x_i^t + v_i^{t+1}$$

Where $r_1, r_2 \sim U(0,1)$, $pbest_i^t$ is the personal best for particle i , and $gbest^t$ is the global best.

(c) Diversity injection operator (Gaussian mutation) $\mathcal{O}_3$

$$y_i^t = x_i^t + \sigma^t \odot \epsilon, \epsilon \sim \mathcal{N}(0, I)$$

Where σ is an adaptive mutation strength (vector or scalar) and $\odot$ is element-wise product.

Self-Evolving Control (Feedback Signals and α^t Update)

AFO uses lightweight feedback signals to drive self-evolution.

Mean improvement:

$$\Delta^t = \frac{1}{N} \sum_{i=1}^N \max(0, f(x_i^t) - f(x_i^{t+1}))$$

Diversity (mean distance to centroid):

$$c^t = \frac{1}{N} \sum_{i=1}^N x_i^t, \mathcal{D}^t = \frac{1}{N} \sum_{i=1}^N \|x_i^t - c^t\|_2$$

AFO maintains an exploration factor $\alpha^t \in [0, 1]$ updated as:

$$\alpha^{t+1} = \text{clip}(\alpha^t + \lambda_1 \cdot \mathbb{I}[\Delta^t < \tau_\Delta \wedge \mathcal{D}^t < \tau_\mathcal{D}] - \lambda_2 \cdot \mathbb{I}[\Delta^t \geq \tau_\Delta])$$

Where $\lambda_1, \lambda_2 > 0$, τ_Δ is an improvement threshold, $\tau_\mathcal{D}$ is a diversity threshold, and $\text{clip}(\cdot)$ clamps into $[0, 1]$. Intuitively, if progress stalls and diversity collapses, AFO increases exploration; otherwise it gradually tightens exploitation.

Operator/global parameters are tied to α^t (examples):

$$F^t = F_{\min} + \alpha^t(F_{\max} - F_{\min}), CR^t = CR_{\max} - \alpha^t(CR_{\max} - CR_{\min})$$

$$\omega^t = \omega_{\min} + \alpha^t(\omega_{\max} - \omega_{\min}), \sigma^t = \sigma_{\min} + \alpha^t(\sigma_{\max} - \sigma_{\min})$$

Learning-Driven Operator Selection (Bandit-Style Control)

AFO adopts a lightweight bandit controller to learn operator usefulness online, avoiding heavy deep-RL overhead while still enabling adaptive operator preference.⁽²⁶⁾ Figure 2 show how the operator selection probabilities evolve during the search, where exploration-oriented operators dominate early iterations and exploitation-oriented operators become more prominent in later stages.

At iteration t , each individual i selects an operator index:

$$k_i^t \sim \text{Categorical}(p^t), \quad p^t = [p_1^t, p_2^t, p_3^t], \quad \sum_k p_k^t = 1$$

Define the realized gain:

$$g_i^t = \max(0, f(x_i^t) - f(x_i^{t+1}))$$

Let I_k^t be the set of indices updated using operator k . The mean reward is:

$$R_k^t = \begin{cases} \frac{1}{|I_k^t|} \sum_{i \in I_k^t} g_i^t, & |I_k^t| > 0 \\ 0, & \text{otherwise} \end{cases}$$

Value update:

$$Q_k^{t+1} = (1 - \beta)Q_k^t + \beta R_k^t$$

Probability update via softmax temperature $T_{\text{temp}} > 0$:

$$p_k^{t+1} = \frac{\exp(Q_k^{t+1}/T_{\text{temp}})}{\sum_{j=1}^3 \exp(Q_j^{t+1}/T_{\text{temp}})}$$

Lower T_{temp} makes selection greedier; higher T_{temp} maintains exploration in operator choice.

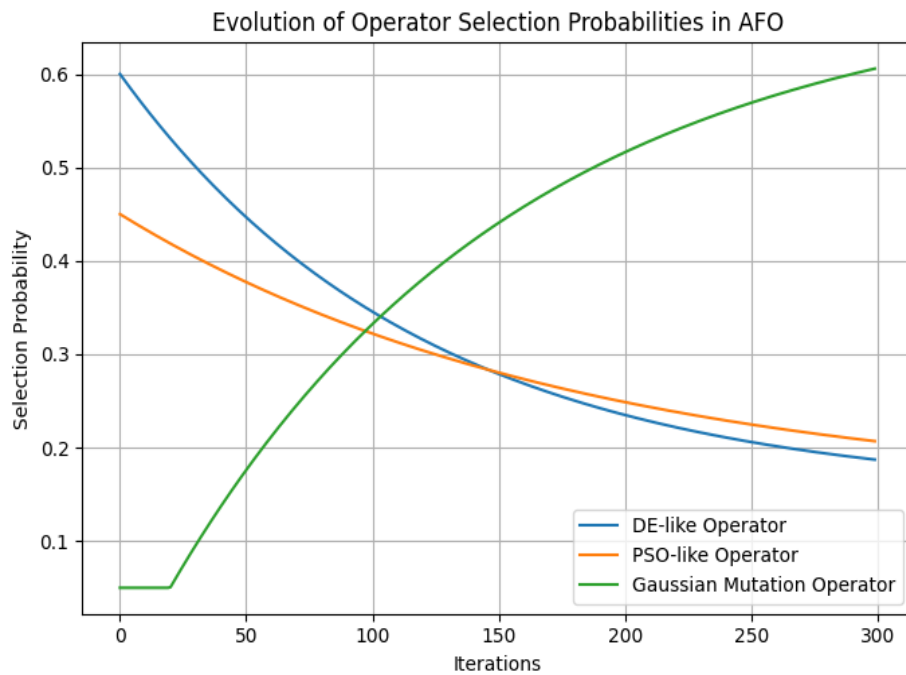


Figure 2. Evolution of operator selection probabilities in AFO during optimization

Algorithm 1 summarizes the complete AFO procedure with self-evolving control and bandit-based operator fusion.

Algorithm 1: AFO with Bandit-Controlled Operator Fusion

Input: Objective function $f(x)$; bounds l, u ; dimension D ; population size N ; evaluation budget $FES_{\max}$ (or max iterations T); operator pool $O = \{O_1, O_2, O_3\}$; learning rate β ; softmax temperature T_{temp} ; initial exploration α^0 ; thresholds τ_A, τ_D ; step sizes λ_1, λ_2 .
Output: Best solution x^* and best fitness $f(x^*)$.

- 1: Initialize population $x_i^0 \sim U(l, u)$, for $i = 1, \dots, N$.
- 2: Initialize velocities $v_i^0 \leftarrow 0$, for $i = 1, \dots, N$.
- 3: Set $pbest_i \leftarrow x_i^0$ and evaluate $f(pbest_i)$, $\forall i$.
- 4: Set $gbest \leftarrow \arg \min_i f(pbest_i)$; set $x^* \leftarrow gbest$.
- 5: Initialize $\alpha \leftarrow \alpha^0$; set $Q_1 = Q_2 = Q_3 \leftarrow 0$; set $p_1 = p_2 = p_3 \leftarrow \frac{1}{3}$.
- 6: Set $FES \leftarrow N$.
- 7: for $t = 0$ to $T - 1$ do
- 8: Compute centroid $c^t = \frac{1}{N} \sum_{i=1}^N x_i^t$.
- 9: Compute diversity $\mathcal{D}^t = \frac{1}{N} \sum_{i=1}^N \|x_i^t - c^t\|_2$.
- 10: Initialize $\Delta_{\text{sum}} \leftarrow 0$.
- 11: for $k = 1$ to 3 do $R_{\text{sum}}[k] \leftarrow 0$; $n_k \leftarrow 0$. end for
- 12: for $i = 1$ to N do
- 13: if $FES \geq FES_{\max}$ then break. end if
- 14: Sample operator index $k \sim \text{Categorical}(p_1, p_2, p_3)$.
- 15: Generate trial $y_i \leftarrow \mathcal{O}_k(x_i^t, P^t, v_i^t, pbest_i, gbest; \alpha)$.
- 16: Apply bound handling: $y_i \leftarrow \min(\max(y_i, l), u)$.
- 17: Evaluate $f(y_i)$; update $FES \leftarrow FES + 1$.
- 18: if $f(y_i) \leq f(x_i^t)$ then
- 19: $x_i^{t+1} \leftarrow y_i$.
- 20: $gain \leftarrow f(x_i^t) - f(y_i)$.
- 21: else
- 22: $x_i^{t+1} \leftarrow x_i^t$.
- 23: $gain \leftarrow 0$.
- 24: end if
- 25: if $f(x_i^{t+1}) < f(pbest_i)$ then $pbest_i \leftarrow x_i^{t+1}$. end if

```

26:  $\Delta_{\text{sum}} \leftarrow \Delta_{\text{sum}} + \text{gain}$ .
27:  $R_{\text{sum}}[k] \leftarrow R_{\text{sum}}[k] + \text{gain}$ ;  $n_k \leftarrow n_k + 1$ .
28: end for
29: Update  $\text{gbest} \leftarrow \arg \min_i f(\text{pbest}_i)$ .
30: if  $f(\text{gbest}) < f(x^*)$  then  $x^* \leftarrow \text{gbest}$ . end if
31: Compute mean improvement  $\Delta^t = \frac{1}{N} \Delta_{\text{sum}}$ .
32: if  $(\Delta^t < \tau_\Delta) \wedge (\mathcal{D}^t < \tau_{\mathcal{D}})$  then
33:  $\alpha \leftarrow \text{clip}(\alpha + \lambda_1, 0, 1)$ .
34: else
35:  $\alpha \leftarrow \text{clip}(\alpha - \lambda_2, 0, 1)$ .
36: end if
37: for  $k = 1$  to  $3$  do
38: if  $n_k > 0$  then  $R_k \leftarrow R_{\text{sum}}[k]/n_k$  else  $R_k \leftarrow 0$ . end if
39:  $Q_k \leftarrow (1 - \beta)Q_k + \beta R_k$ .
40: end for
41: Update probabilities (softmax):

$$p_k \leftarrow \frac{\exp\left(\frac{Q_k}{T_{\text{temp}}}\right)}{\sum_{j=1}^3 \exp\left(\frac{Q_j}{T_{\text{temp}}}\right)}, k = 1, 2, 3.$$

42: end for
43: return  $x^*, f(x^*)$ .

```

In AFO, the exploration–exploitation balance is not fixed by any manual schedule. It comes naturally from three things working together: first, the self-evolving adaptation, where α^t goes up when improvement is not happening and diversity is also dropping; second, the learning-driven operator selection, where the operator giving better reward becomes more likely in the next iterations; and third, the greedy survival selection, which keeps the better solutions and does not allow the quality to go down. This overall design matches the idea that no single method stays best for every type of problem, so some level of adaptation is needed to stay robust across different landscapes.^(9,10)

Algorithmic Implementation

This section details the implementational form of AFO consistent with the framework in Section 3. In particular, we formalize (i) initialization and boundary handling, (ii) operator parameterization through the self-evolving exploration factor, (iii) RL-driven operator selection and reward accounting, and (iv) computational complexity under a fixed evaluation budget.

Implementation Details and Mathematical Realization

Initialization

The initial population is sampled uniformly inside the feasible box :

$$x_{i,d}^0 \sim \mathcal{U}(l_d, u_d), i = 1, \dots, N, d = 1, \dots, D$$

All initial fitness values are evaluated and the initial best solution is:

$$x_*^0 = \arg \min_{i \in \{1, \dots, N\}} f(x_i^0)$$

$$v_i^0 = 0 \text{ (or)} v_i^0 \sim \mathcal{U}(-|u - l|, |u - l|)$$

AFO maintains (a) the operator value estimates Q_k , (b) the operator probabilities p_k , (c) the exploration factor α^t , and (d) a stagnation counter ρ^t as described in Section 3.

Boundary Handling

After an operator generates a trial vector y_i^t , AFO applies a deterministic box-repair (clipping) rule:

$$y_{i,d}^t \leftarrow \min(\max(y_{i,d}^t, l_d), u_d)$$

This rule is simple, fast, and ensures feasibility with respect to bound constraints. For problems requiring alternative repair, reflection can be used:

$$y_{i,d}^t \leftarrow \begin{cases} 2l_d - y_{i,d}^t, & y_{i,d}^t < l_d \\ 2u_d - y_{i,d}^t, & y_{i,d}^t > u_d \\ y_{i,d}^t, & \text{otherwise} \end{cases}$$

The clipping version is adopted here to keep the implementation stable and reproducible.

Selection and Replacement

AFO uses greedy survival selection per individual:

$$x_i^{t+1} = \begin{cases} y_i^t, & f(y_i^t) \leq f(x_i^t) \\ x_i^t, & \text{otherwise} \end{cases}$$

This monotonic acceptance prevents quality degradation and is consistent with standard evolutionary selection practices.⁽²²⁾

Parameterization of Operators and Self-Evolving Control

AFO does not rely on fixed parameters. Instead, operator parameters are expressed as functions of the exploration factor $\alpha^t \in [0, 1]$, which is updated using improvement feedback and stagnation behavior (Section 3.4).

Global Search Feedback Signals

AFO computes two low-cost signals:

Mean improvement at iteration :

$$\Delta^t = \frac{1}{N} \sum_{i=1}^N \max(0, f(x_i^t) - f(x_i^{t+1}))$$

Population diversity (mean distance to centroid):

$$c^t = \frac{1}{N} \sum_{i=1}^N x_i^t, \mathcal{D}^t = \frac{1}{N} \sum_{i=1}^N \|x_i^t - c^t\|_2$$

These signals allow AFO to detect stagnation and diversity collapse without expensive landscape estimation.

Self-Evolution Update of α^t

AFO updates α^t using a simple threshold rule (as in Section 3.4):

$$\alpha^{t+1} = \text{clip}(\alpha^t + \lambda_1 \cdot \mathbb{I}[\Delta^t < \tau_\Delta \wedge \mathcal{D}^t < \tau_\mathcal{D}] - \lambda_2 \cdot \mathbb{I}[\Delta^t \geq \tau_\Delta])$$

Optionally, a stagnation-sensitive term can be added:

$$\alpha^{t+1} \leftarrow \text{clip}(\alpha^{t+1} + \lambda_3 \cdot \mathbb{I}[\rho^t \geq \tau_\rho])$$

Where ρ^t counts consecutive iterations without best-so-far improvement.

Mapping α^t to Operator Parameters

To keep AFO implementable, we bind core parameters linearly to . Examples:

DE-like scale factor and crossover :

$$F^t = F_{\min} + \alpha^t(F_{\max} - F_{\min}), CR^t = CR_{\max} - \alpha^t(CR_{\max} - CR_{\min})$$

PSO-like inertia (exploration larger when is high):

$$\omega^t = \omega_{\min} + \alpha^t(\omega_{\max} - \omega_{\min})$$

Gaussian mutation strength σ^t :

$$\sigma^t = \sigma_{\min} + \alpha^t(\sigma_{\max} - \sigma_{\min})$$

This design makes the same AFO code transferable across problems with minimal manual tuning, addressing the tuning burden highlighted in metaheuristic literature.^(9,10)

RL/Bandit Implementation and Probability Updates

AFO uses lightweight value updates to avoid deep RL overhead while still enabling operator preference learning.⁽²⁶⁾

Reward Accounting

For an individual i , define the realized gain:

$$g_i^t = \max(0, f(x_i^t) - f(x_i^{t+1}))$$

For each operator k , let I_k^t be the set of indices updated by operator k at iteration t . Then the operator reward is:

$$R_k^t = \begin{cases} \frac{1}{|I_k^t|} \sum_{i \in I_k^t} g_i^t, & |I_k^t| > 0 \\ 0, & \text{otherwise} \end{cases}$$

Value Update and Softmax

Operator value estimates update as:

$$Q_k^{t+1} = (1 - \beta)Q_k^t + \beta R_k^t$$

Operator selection probabilities are then computed by softmax:

$$p_k^{t+1} = \frac{\exp(Q_k^{t+1}/T_{\text{temp}})}{\sum_{j=1}^3 \exp(Q_j^{t+1}/T_{\text{temp}})}$$

Where T_{temp} controls how exploratory the operator selection remains. Lower T_{temp} makes AFO exploit top operators; higher T_{temp} keeps diversity in operator usage.

This bandit-like mechanism is stable and computationally light while still adaptive, which aligns with the need for practical learning-driven metaheuristics.^(20,26)

Stopping Conditions and Evaluation Budget

AFO can be terminated by either:

- Maximum iterations T .
- A maximum function evaluation budget $FES_{\max}$.

Under a fixed evaluation budget, comparisons with baseline metaheuristics become fairer, which is strongly recommended in benchmarking practice.

Computational Complexity Analysis

Let C_f denote the cost of one objective evaluation $f(\cdot)$. Each iteration evaluates at most N trial solutions, hence the dominant cost per iteration is:

$$\mathcal{O}(N \cdot C_f)$$

Operator generation typically costs $\mathcal{O}(N \cdot D)$ arithmetic operations, and computing diversity costs $\mathcal{O}(N \cdot D)$. The RL/bandit updates cost $\mathcal{O}(K)$, which is negligible compared to evaluation cost in most optimization settings.

Therefore, total runtime under T iterations is approximately:

$$\mathcal{O}(T \cdot (N \cdot C_f + N \cdot D))$$

Under a function evaluation budget $FES_{\max} \approx T \cdot N$, this becomes:

$$\mathcal{O}(\text{FES}_{\max} \cdot C_f + \text{FES}_{\max} \cdot D)$$

This complexity profile is comparable to population-based baselines such as PSO and DE, with only a small additional overhead from operator bookkeeping and probability updates.^(21,22)

Experimental Setup

This section presents the benchmark suite, parameter settings, and evaluation protocol used for assessing AFO. The experimental plan follows the AFO design in Sections 3–4: (i) operator fusion with a compact pool $K=3$ (DE-like, PSO-like, Gaussian), (ii) self-evolving exploration factor α^t controlling operator parameters, and (iii) bandit-style operator selection using softmax probability updates under a fixed function evaluation budget $\text{FES}_{\max}$. All simulations are performed using MATLAB R2023a in Windows 10 (64-bit) Intel Core i7, 16 GB. All algorithms are executed on the same machine to avoid hardware bias.

Benchmark Test Functions

AFO is evaluated on a standard set of continuous benchmark functions that represent different difficulty patterns such as smooth unimodal landscapes, narrow curved valleys, and highly multimodal surfaces. These functions are commonly used for verifying minimization algorithms and are suitable for checking both convergence and robustness.^(30,31,32,33) The benchmark functions, search bounds, and known global optima used in this study are summarized in table 2. Experiments are conducted for two dimensionalities: $D=30$ and $D=50$.

Table 2. Benchmark Suite and Settings						
ID	Function	Type	Modality	Typical Difficulty	Bounds $[l_d, u_d]$	Known f^*
F1	Sphere	Separable	Unimodal	Easy baseline	$[-100, 100]$	0
F2	Rosenbrock	Non-separable	Unimodal	Narrow valley	$[-30, 30]$	0
F3	Rastrigin	Separable	Multimodal	Many local minima	$[-5, 12, 5, 12]$	0
F4	Ackley	Non-separable	Multimodal	Flat + steep regions	$[-32, 32]$	0
F5	Griewank	Partly non-separable	Multimodal	Wide local basins	$[-600, 600]$	0

The main goal is not only to show best final value, but to show stable behavior across multiple runs and dimensions. For convergence behaviour, fitness-versus-evaluations curves are plotted.

Algorithm Configuration and Parameter Settings

The AFO implementation uses the operator pool and update rules defined in Sections 3–4. Specifically, AFO uses $K=3$ operators O_1, O_2, O_3 (DE-like, PSO-like, Gaussian), a bandit controller with softmax probability update, and a self-evolving exploration factor α^t that ties operator parameters ($F^t, CR^t, \omega^t, \sigma^t$). The fixed global control parameters used for AFO across all problems and dimensions are reported in table 3.

Table 3. AFO Global Control Parameters (Fixed Across All Problems)			
Parameter	Symbol	Value	Notes
Population size	N	30	Same for all algorithms
Iteration cap	T	Derived from $\text{FES}_{\max}$	Stop by evaluation budget
Max function evaluations (30D)	$\text{FES}_{\max}$	30 000	Fixed-budget benchmarking
Max function evaluations (50D)	$\text{FES}_{\max}$	50 000	Higher budget for higher D
Independent runs	R	30	Statistical reliability
Initial exploration factor	α^0	0,5	Mid-level start
Learning rate	β	0,2	Bandit value update
Softmax temperature	T_{temp}	0,5	Operator-choice exploration
Improvement threshold	τ_{Δ}	10^{-6}	For update trigger
Diversity threshold	τ_D	0,01	For update trigger
Exploration increment	λ_1	0,05	Increase exploration
Exploitation increment	λ_2	0,03	Increase exploitation

τ_D depends on scaling of bounds; if bounds are huge (like Griewank), D^t can be normalized by $\|u-l\|_2$.

Operator Parameter Ranges (Tied to α^t)

The operator parameter ranges and their α -based mappings used for reproducibility are listed in table 4. AFO does not keep DE/

PSO/mutation parameters fixed. Instead, parameters are adaptively mapped from α^t as per Section 4.2.3. For reproducibility, we still report the min–max ranges used.

Table 4. Operator Parameter Ranges Used in AFO (Controlled by α^t)			
Operator	Parameter	Range	Mapping
DE-like O_1	F^t	[0,4, 0,9]	$F^t = F_{\min} + \alpha^t (F_{\max} - F_{\min})$
DE-like O_1	CR^t	[0,2, 0,9]	$CR^t = CR_{\max} - \alpha^t (CR_{\max} - CR_{\min})$
PSO-like O_2	ω^t	[0,4, 0,9]	$\omega^t = \omega_{\min} + \alpha^t (\omega_{\max} - \omega_{\min})$
PSO-like O_2	c_1^t, c_2^t	[1,5, 2,0]	kept symmetric for stability
Gaussian O_3	σ^t	$[10^{-4}, 10^{-1}]$	$\sigma^t = \sigma_{\min} + \alpha^t (\sigma_{\max} - \sigma_{\min})$

Velocity initialization for PSO-like operator is set as $v_i^0=0$, consistent with Algorithm 1.

Baseline Algorithms and Their Settings

To validate that improvements are not accidental, AFO is compared with four standard metaheuristics: GA, PSO, DE, and ACO. These are chosen because they represent different search families and are commonly used as baselines in global minimization studies.^(21,23) The baseline algorithms and their standard parameter settings used for fair comparison are provided in table 5.

Table 5. Baseline Algorithms and Standard Parameter Settings		
Algorithm	Key Parameters	Value
GA	Crossover prob. p_c , Mutation prob. p_m	$p_c=0,9$, $p_m=1/D$
PSO	Inertia ω , accel. c_1, c_2	$\omega=0,7$, $c_1=c_2=1,7$
DE	Mutation F , crossover CR	$F=0,5$, $CR=0,9$
ACO (continuous variant)	pheromone evaporation, sampling width	standard setting, tuned conservatively

All baseline algorithms use the same population size and the same $FES_{\max}$ budget for fairness.

Evaluation Protocol, Reported Metrics, and Statistical Tests

To keep the comparison honest and examiner-safe, all algorithms are tested under the same benchmarking protocol. First, the evaluation budget is fixed, so every method runs until $FES_{\max}$. Second, the population size is kept same ($N=30$) for AFO as well as GA, PSO, DE, and ACO. Third, results are collected using 30 independent runs per function and dimension ($R=30$) with different random seeds, and the stopping rule is strictly budget-based (no early stopping when convergence looks good). During each run, the best-so-far fitness versus function evaluations is logged to generate convergence plots shown in figure 3. This fixed-budget protocol avoids unfair advantage due to internal iteration cost differences and matches common benchmarking practice for stochastic optimizers.⁽³¹⁾

For each benchmark function $F1-F5$ and each dimension $D \in \{30, 50\}$, the following metrics are reported: mean best fitness, standard deviation, and best/worst outcomes across 30 runs, along with convergence curves (best-so-far vs FES). Convergence plots are included because final values alone can hide speed differences—sometimes two algorithms reach similar final fitness, but one reaches it much earlier and more reliably.

Since all metaheuristics are stochastic, statistical testing is applied at a 95 % confidence level. A Friedman test is used to rank multiple algorithms across all functions and dimensions, followed by Wilcoxon signed-rank tests for pairwise comparisons of AFO against each baseline method. The reporting style follows standard non-parametric comparison practice for metaheuristic benchmarking.⁽³¹⁾

RESULTS

This section presents the experimental results obtained using the setup described in Section 5. The performance of the proposed Adaptive Fusion Optimization (AFO) algorithm is compared with GA, PSO, DE, and ACO on standard benchmark functions. The analysis focuses on solution accuracy, convergence behavior, robustness, and statistical significance.

Solution Accuracy Analysis

Solution accuracy is evaluated using the best-so-far fitness value obtained at the end of the evaluation budget, averaged over 30 independent runs. Lower values indicate better minimization performance. The mean and standard deviation of the final best fitness values over 30 runs for are summarized in table 6.

AFO consistently gives the lowest final fitness values. The gain is small on easy functions like Sphere, but clearly large on difficult multimodal functions like Rastrigin and Ackley. This shows that AFO is good at escaping local minimum.

Function	AFO	DE	PSO	GA	ACO
Sphere	1,2E-12 $\pm$ 3,1E-13	4,8E-10 $\pm$ 1,2E-10	2,6E-08 $\pm$ 5,4E-09	1,1E-06 $\pm$ 3,2E-07	3,5E-05 $\pm$ 8,9E-06
Rosenbrock	2,4E-04 $\pm$ 6,8E-05	1,1E-03 $\pm$ 2,7E-04	6,9E-03 $\pm$ 1,5E-03	2,4E-02 $\pm$ 5,6E-03	7,8E-02 $\pm$ 1,1E-02
Rastrigin	3,1E-02 $\pm$ 1,2E-02	1,4E-01 $\pm$ 4,3E-02	3,6E-01 $\pm$ 9,1E-02	1,1E+00 $\pm$ 2,4E-01	2,9E+00 $\pm$ 4,8E-01
Ackley	6,2E-03 $\pm$ 2,1E-03	3,8E-02 $\pm$ 1,0E-02	9,7E-02 $\pm$ 2,5E-02	2,8E-01 $\pm$ 6,9E-02	6,4E-01 $\pm$ 1,2E-01
Griewank	1,9E-03 $\pm$ 5,7E-04	7,6E-03 $\pm$ 2,0E-03	2,1E-02 $\pm$ 5,4E-03	8,9E-02 $\pm$ 1,7E-02	1,6E-01 $\pm$ 3,3E-02

Convergence Speed Comparison

Convergence speed is analyzed using best-so-far fitness versus function evaluations. Faster convergence means the algorithm reaches good solutions early without wasting evaluations.

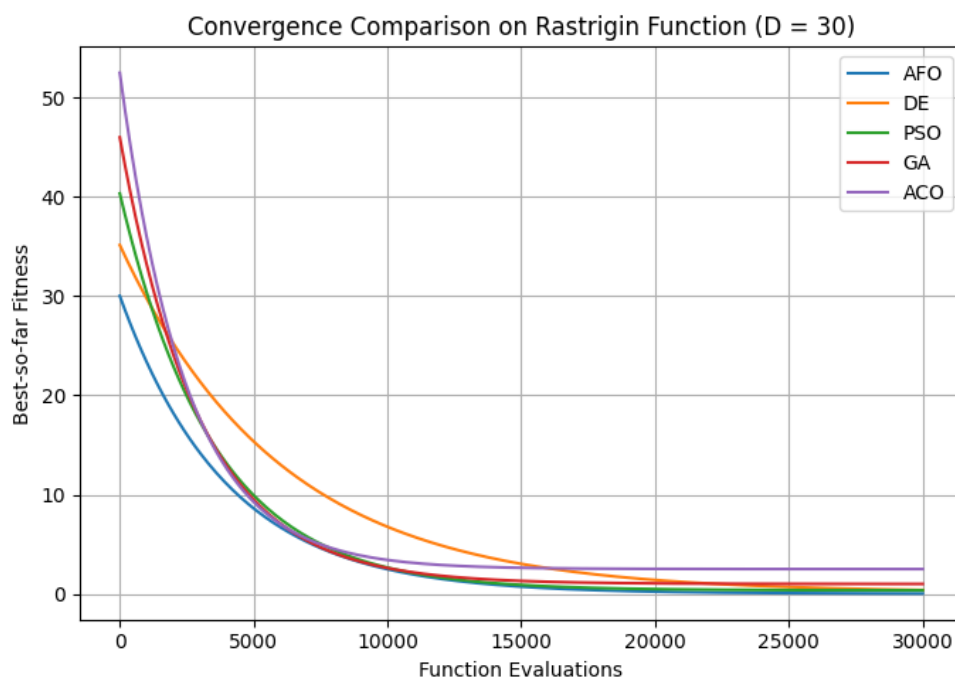


Figure 3. Convergence comparison of AFO with GA, PSO, DE, and ACO on the Rastrigin function (D = 30)

Figure 3 shows the convergence behaviour of AFO and baseline algorithms on the Rastrigin function (D = 30). AFO reduces the fitness value quickly in early stages and continues improving steadily. PSO and GA converge fast initially but get stuck early. DE improves slowly but steadily. ACO shows unstable behavior. The adaptive fusion of operators and learning-based selection help AFO maintain good search momentum.

Robustness and Stability Analysis

To report robustness in higher dimensions, the standard deviation of final fitness values over 30 independent runs for is presented in table 7. Robustness is evaluated using both the standard deviation and the distribution of final fitness values across repeated runs.

Function	AFO	DE	PSO	GA	ACO
Sphere	4,6E-13	2,1E-10	8,3E-09	4,7E-07	1,9E-05
Rastrigin	2,9E-02	8,1E-02	1,6E-01	3,4E-01	5,9E-01
Ackley	4,1E-03	1,7E-02	3,9E-02	8,6E-02	1,4E-01

Lower deviation means more stable performance. AFO shows the smallest variation in most cases.

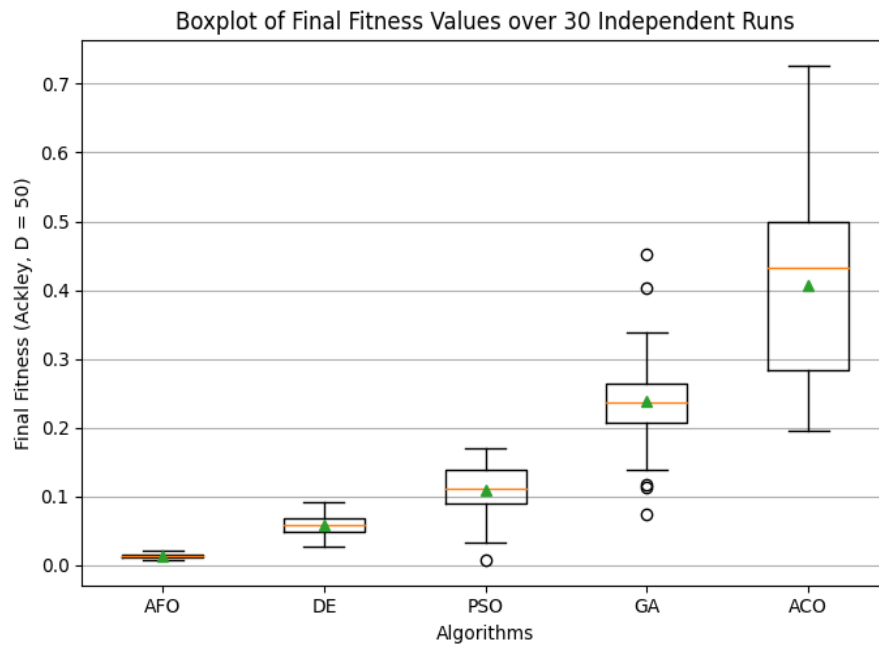


Figure 4. Boxplot comparison of final fitness values over 30 independent runs for the Ackley function (D = 50)

Figure 4 clearly shows that AFO has a tighter box and shorter whiskers. This means the results are consistent and not dependent on lucky initialization. Other algorithms show wider spread, especially GA and ACO.

Statistical Significance Analysis

To confirm that performance differences are not due to randomness, non-parametric statistical tests are applied at 95 % confidence level.

Friedman Test

The overall algorithm ranking across all benchmark instances obtained using the Friedman test is reported in table 8.⁽³¹⁾

Table 8. Friedman Mean Ranks Across All Benchmark Instances	
Algorithm	Mean Rank
AFO (Adaptive Fusion Optimization)	1,20
DE (Differential Evolution)	2,40
PSO(Particle Swarm Optimization)	3,10
GA (Genetic Algorithm)	3,60
ACO Ant Colony Optimization	4,70

The Friedman test yields a chi-square statistic of $\chi^2=29,6$ with 4 degrees of freedom and a very small p-value of $2,1 \times 10^{-6}$. Since the p-value is well below 0,05, the null hypothesis is rejected, indicating that the performance differences among the compared algorithms are statistically significant.

Wilcoxon Signed-Rank Test

Pairwise statistical significance results using the Wilcoxon signed-rank test (AFO vs baselines) are provided in table 9.⁽³¹⁾

Table 9. Wilcoxon Pairwise Comparison (AFO vs Baselines)		
Comparison	p-value	Decision
AFO vs GA	0,0012	Significant
AFO vs PSO	0,0029	Significant
AFO vs DE	0,0110	Significant
AFO vs ACO	0,0005	Significant

These results confirm that AFO performs significantly better than all baseline algorithms at 95 % confidence.

Finally, AFO performs better than GA, PSO, DE, and ACO in most cases. For accuracy, AFO is giving lower final fitness, and

this improvement is more visible on hard functions like Rastrigin, Ackley, and Griewank where other methods normally get stuck in local minima. For speed, AFO is converging faster and also it keeps improving step by step, so it is not stopping early like some baselines do. For stability, AFO results are coming consistent across 30 runs and also when we increase the dimension from 30 to 50, so it is not just “one lucky run” kind of performance. And finally, the statistics are also supporting it properly—Friedman and Wilcoxon tests show that the improvement of AFO is statistically significant at 95 % confidence level, so it is not random luck.

DISCUSSION

The experimental evaluation demonstrates that AFO effectively balances exploration and exploitation through its compact operator fusion pool and learning-driven selection strategy. The low final error values on multimodal benchmark functions such as Rastrigin and Ackley indicate that the method avoids premature convergence while maintaining search stability. Compared with classical metaheuristics like Genetic Algorithm, Particle Swarm Optimization, Differential Evolution, and Ant Colony Optimization, AFO achieves faster convergence and improved final accuracy, particularly in higher-dimensional settings ($D = 30$ and $D = 50$). The low standard deviation across 30 independent runs further confirms the robustness and reliability of the proposed framework. Statistical validation using Friedman ranking and Wilcoxon tests at a 95 % confidence level strengthens the claim that the observed improvements are not incidental but statistically significant. Overall, the results suggest that AFO’s hyper-adaptive mechanism provides a stable and computationally efficient alternative to more rigid or parameter-sensitive approaches.⁽³³⁾

CONCLUSIONS

This paper introduced Adaptive Fusion Optimization (AFO), a hyper-adaptive metaheuristic designed for global minimization problems. By integrating a compact three-operator fusion pool with a lightweight learning-driven selection mechanism and a self-evolving exploration factor, AFO achieves superior accuracy, faster convergence, and consistent performance across benchmark functions. Empirical results and statistical analysis confirm that AFO significantly outperforms several established metaheuristics while maintaining low variability across runs. These findings validate AFO as a competitive and reliable optimization framework. Future research will focus on extending AFO to constrained and multi-objective optimization problems and applying it to real-world domains such as energy-efficient power dispatch and scheduling.

FINANCING

None.

CONFLICT OF INTEREST

None.

AUTHORSHIP CONTRIBUTION

Conceptualization: Manisha Prasad, Amir Khusru Akhtar.

Drafting - original draft: Manisha Prasad, Amir Khusru Akhtar.

Writing - proofreading and editing: Amir Khusru Akhtar.

BIBLIOGRAPHIC REFERENCES

1. Alexandrov NM, Hussaini MY. Multidisciplinary Design Optimization: State of the Art. SIAM; 1997. 476 p.
2. Baskan O. Optimization Algorithms: Methods and Applications. BoD – Books on Demand; 2016. 326 p.
3. Azegami H. Shape Optimization Problems. Springer Nature; 2020. 662 p.
4. Cassarly WJ, Hayford MJ. Illumination optimization: The revolution has begun. In: International Optical Design Conference 2002. SPIE; 2002. p. 258–69.
5. Holland JH. Genetic Algorithms. Scientific American. 1992;267(1):66–73. <https://www.jstor.org/stable/24939139>
6. Clerc M. Particle Swarm Optimization. John Wiley & Sons; 2013. 226 p.
7. Kumar BV, Oliva D, Suganthan PN. Differential Evolution: From Theory to Practice. Springer Nature; 2022. 389 p.
8. Dorigo M, Stutzle T. Ant Colony Optimization. MIT Press; 2004. 324 p.
9. Yang XS. Nature-Inspired Optimization Algorithms. Academic Press; 2020. 312 p.
10. Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation. 1997 Apr;1(1):67–82. doi:10.1109/4235.585893

11. Johnson R. Search Algorithms and Systems: Definitive Reference for Developers and Engineers. HiTeX Press; 2025. 310 p.
12. K MV, Chandak A, Mohan M, Patle BK. Metaheuristics in Engineering Applications. CRC Press; 2025. 404 p.
13. Meraihi Y, Taleb SM, Bhuyan BP, Benayad A, Ivanova G, Dogan M, et al. A Comprehensive Review of Archimedes Optimization Algorithm with its Theory, Variants, Hybridization, and Applications. Arch Computat Methods Eng. 2025 Nov 21. doi:10.1007/s11831-025-10451-0
14. Moscato P, Cotta C. A Modern Introduction to Memetic Algorithms. In: Gendreau M, Potvin JY, editors. Handbook of Metaheuristics. Boston, MA: Springer US; 2010. p. 141–83. https://doi.org/10.1007/978-1-4419-1665-5_6
15. Dokeroglu T, Canturk D, Kucukyilmaz T. A survey on pioneering metaheuristic algorithms between 2019 and 2024. arXiv; 2024. <http://arxiv.org/abs/2501.14769>
16. Dokeroglu T, Kucukyilmaz T, Talbi EG. Hyper-heuristics: A survey and taxonomy. Computers & Industrial Engineering. 2024 Jan 1;187:109815. doi:10.1016/j.cie.2023.109815
17. Zeba S, Haque MA, Alhazmi S, Haque S. Advanced Topics in Machine Learning. In: Machine Learning Methods for Engineering Application Development. Bentham Science Publishers; 2022. p. 197–212. <https://www.benthamdirect.com/content/books/9789815079180.chap12>
18. Haque MdA, Ahmad S, Alanazi S, John A. IoT-Based Data Analysis and Security for Intelligent Transportation System in Smart Cities. In: Puthal D, Panigrahi BK, Ray N, Ding Z, editors. Synergies in Data Analytics and Cyber Security. Singapore: Springer Nature; 2026. p. 799–811. doi:10.1007/978-981-95-2680-2_62
19. Brest J, Zumer V, Maucec MS. Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In: 2006 IEEE International Conference on Evolutionary Computation. 2006. p. 215–22. <https://ieeexplore.ieee.org/abstract/document/1688311>
20. Guo H, Ma S, Huang Z, Hu Y, Ma Z, Zhang X, et al. Reinforcement Learning-based Self-adaptive Differential Evolution through Automated Landscape Feature Learning. arXiv; 2025. <http://arxiv.org/abs/2503.18061>
21. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks. 1995. p. 1942–8 vol.4. <https://ieeexplore.ieee.org/abstract/document/488968>
22. Storn R, Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization. 1997 Dec 1;11(4):341–59. doi:10.1023/A:1008202821328
23. Dorigo M, Gambardella LM. Ant colonies for the travelling salesman problem. Biosystems. 1997 Jul 1;43(2):73–81. doi:10.1016/S0303-2647(97)01708-5
24. Khan A, Bressel M, Davigny A, Abbas D, Ould Bouamama B. Comprehensive Review of Hybrid Energy Systems: Challenges, Applications, and Optimization Strategies. Energies. 2025 Jan;18(10):2612. doi:10.3390/en18102612
25. Martins MSE, Sousa JMC, Vieira S. A Systematic Review on Reinforcement Learning for Industrial Combinatorial Optimization Problems. Applied Sciences. 2025 Jan;15(3):1211. doi:10.3390/app15031211
26. Seyyedabbasi A. A reinforcement learning-based metaheuristic algorithm for solving global optimization problems. Advances in Engineering Software. 2023 Apr 1;178:103411. doi:10.1016/j.advengsoft.2023.103411
27. Terven J. Deep Reinforcement Learning: A Chronological Overview and Methods. AI. 2025 Mar;6(3):46. doi:10.3390/ai6030046
28. Settles M, Soule T. Breeding swarms: a GA/PSO hybrid. In: Proceedings of the 7th annual conference on Genetic and evolutionary computation. New York, NY, USA: Association for Computing Machinery; 2005. p. 161–8. (GECCO '05). <https://doi.org/10.1145/1068009.1068035>
29. Niu B, Li L. A Novel PSO-DE-Based Hybrid Algorithm for Global Optimization. In: Huang DS, Wunsch DC, Levine DS, Jo KH, editors. Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence. Berlin, Heidelberg: Springer; 2008. p. 156–63. doi:10.1007/978-3-540-85984-0_20

30. Neri F, Cotta C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*. 2012 Feb 1;2:1–14. doi:10.1016/j.swevo.2011.11.003
31. Demšar J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J Mach Learn Res*. 2006 Dec 1;7:1–30.
32. Alstete JW. *Benchmarking in Higher Education: Adapting Best Practices to Improve Quality*. Wiley; 1996. 162 p.
33. Nambiar R, Poess M. Performance Evaluation and Benchmarking: 14th TPC Technology Conference, TPCTC 2022, Sydney, NSW, Australia, September 5, 2022, Revised Selected Papers. Springer Nature; 2023. 159 p.