Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(10): 7619–7650.
DOI: 10.3934/era.2026329
Received: 14 June 2026
Revised: 06 August 2026
Accepted: 14 August 2026
Published: 04 September 2026

Research article

A dynamic self-adaptive optimization framework integrating the metaheuristic strategy for the non-convex stochastic optimization problem

Weiyi Jin[1], **Sha Lu**[1,2,*] **and Libin Liu**[1,2]

[1] Center for Applied Mathematics of Guangxi, Nannning Normal University, Nanning 530100, China
[2] Guangxi Key Lab of Human-machine Interaction and Intelligent Decision, Nanning 530100, China

* **Correspondence:** Email: lusha@nnnu.edu.cn.

Abstract: As a critical branch of optimization theory, adaptive optimization algorithms mitigate the limitations of classical gradient descent, such as slow convergence and vulnerability to local optima, particularly in high-dimensional, non-convex environments. This is achieved by dynamically adjusting learning rates and per-parameter update rules. By integrating metaheuristic strategies with adaptive gradients, a dynamic self-adaptive optimization framework (DSOF) is proposed in this paper. The DSOF introduces a history-informed second-order momentum mechanism, which leverages recent gradient statistics to improve robustness under non-stationary objectives. Additionally, it maintains population diversity through evolutionary crossover and mutation operations to prevent premature convergence, and adaptively contracts or expands parameter bounds to balance exploration and exploitation. Experimental results demonstrated that, under the evaluated settings, the DSOF improves optimization stability by reducing the gradient-variance fluctuation coefficient from 0.418 to 0.382 in the modified national institute of standards and technology classification task, achieves a 100% global optimum hit rate on the 50-dimensional Ackley function, and reduces NAS-Bench-201 search time to 25 hours with only 1.02 GPU-days. These results verify the empirical effectiveness of the DSOF in improving stability, global exploration capability, and computational efficiency.

Keywords: adaptive optimization algorithms; metaheuristic strategy; population diversity strategy; second-order momentum; dynamic adjustment

1. Introduction

Non-convex stochastic optimization plays a fundamental role in a wide range of applications, including deep neural network training, reinforcement learning, autonomous system control, neural architecture search, and unmanned aerial vehicle (UAV) path planning. In these applications, the objective function is typically high-dimensional, non-convex, and evaluated based on noisy

observations or stochastic samples. Consequently, the optimization process is often challenged by local optima, saddle points, flat regions, stochastic gradient noise, and temporal variations in data distributions or operating environments. Such factors may lead to unstable parameter updates, slow convergence, and premature convergence to suboptimal solutions.

Let $\theta \in \mathbb{R}^d$ denote the parameter vector to be optimized, and let $\zeta \sim \mathcal{D}$ represent a random sample, perturbation, or environment-induced variation. The optimization objective is formulated as

$$\min_{\theta \in \Theta} \mathbb{E}_{\zeta \sim \mathcal{D}} \left[f(\theta; \zeta) \right], \tag{1.1}$$

where $f(\theta; \zeta)$ is a potentially non-convex stochastic loss function, $\Theta \subseteq \mathbb{R}^d$ is the feasible parameter domain, and $\mathcal{D}$ denotes the underlying data or environment distribution. In practical applications, the expected objective is commonly approximated using a finite dataset:

$$\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} f_i(\theta), \tag{1.2}$$

where $f_i(\theta)$ denotes the loss associated with the i-th sample.

Solving the formulated problem is challenging because stochastic gradient noise, multimodal objective landscapes, and temporal non-stationarity may interact throughout the optimization process. The observed stochastic gradient $g_t = \nabla f(\theta_t; \zeta_t)$ may exhibit high variance or heavy-tailed behavior and therefore provide an unstable approximation of the underlying descent direction, leading to oscillatory updates and degraded optimization stability. Meanwhile, highly non-convex landscapes often contain numerous local optima and saddle regions, making optimizers that rely solely on local gradient information susceptible to premature convergence. In non-stationary applications, changes in data distributions, operating environments, or task requirements may further alter the loss landscape over time. An effective optimizer should therefore respond jointly to gradient uncertainty, search diversity, and optimization progress.

Classical gradient descent and stochastic gradient descent provide efficient local updates but may converge slowly in ill-conditioned regions and remain sensitive to saddle points, flat areas, and stochastic perturbations [1–3]. Adaptive gradient methods, such as AdaGrad, RMSProp, Adam, AdamW, and related variants, improve local optimization by dynamically adjusting parameter-wise learning rates and incorporating momentum information [4–6]. Nevertheless, adaptive gradient methods mainly exploit local gradient statistics. AdaGrad may suffer from excessively decaying learning rates, whereas RMSProp and Adam can remain sensitive to abrupt changes in gradient variance and may converge to unfavorable regions in highly multimodal landscapes [7, 8]. In addition, exponentially weighted estimators emphasize recent gradients and may not adequately characterize gradient variance under strongly noisy or non-stationary conditions.

Evolutionary and metaheuristic algorithms address global exploration from a different perspective. Genetic algorithms, particle swarm optimization, ant colony optimization, the covariance matrix adaptation evolution strategy (CMA-ES), and natural evolution strategies (NESs) explore multiple candidate solutions and can escape local optima more effectively than purely gradient-based methods [9, 10]. However, population-based search may incur substantial computational and memory costs, especially in high-dimensional learning tasks, and may exhibit slow local refinement when accurate gradient information is available. Restart-based strategies, such as stochastic gradient

descent with warm restarts, periodically increase the learning rate to renew exploration [11]. Although effective in many applications, restart-based strategies generally follow predefined schedules rather than adapting directly to the observed gradient variance or optimization progress.

Existing methods often emphasize stable local descent, population-based global exploration, or dynamic search control separately. Coordinating all three capabilities within a unified optimizer remains challenging. An integrated optimizer should suppress noise-induced oscillations during local updates, maintain sufficient solution diversity to escape unfavorable regions, and adapt the search range according to the observed optimization state. Such requirements motivate an integrated approach that combines gradient-based learning, evolutionary exploration, and progress-dependent search-space adjustment.

To address the identified limitations, a dynamic self-adaptive optimization framework (DSOF) is proposed by integrating three complementary mechanisms. The dynamic second-order momentum mechanism (DSMM) estimates the mean and variance of recent gradients using a sliding-window history and regulates the learning rate according to the resulting uncertainty information. The hybrid metaheuristic strategy for diversity-enhanced exploration (HMSDE) periodically incorporates crossover, mutation, and elite preservation into the gradient-based optimization process, thereby maintaining population diversity and generating candidate solutions beyond the current local region. The autonomous parameter space adjustment mechanism (APSA) modifies the search boundaries according to the historical improvement of the best objective value, expanding the search range when stagnation is detected and contracting the search range when stable progress is observed.

The three mechanisms serve distinct but complementary purposes. The DSMM primarily addresses stochastic gradient noise and update instability, the HMSDE improves global exploration in multimodal landscapes, and the APSA adapts the exploration range according to stagnation and optimization progress. Unlike the CMA-ES and NES, which primarily optimize an explicit population-level search distribution, the DSOF retains direct stochastic-gradient updates and invokes evolutionary operations periodically. Unlike restart-based methods that follow a predetermined learning-rate schedule, the DSOF adjusts the learning rate and parameter bounds using observed gradient variance and optimization progress. Therefore, the DSOF combines variance-adaptive local optimization with heuristic exploration and dynamic search-space adjustment. The effectiveness of the DSOF is evaluated empirically under the considered experimental settings, and no formal convergence guarantee is claimed for the complete framework.

The principal contributions of the present work are summarized as follows:

- A unified optimization framework is proposed by integrating variance-aware gradient regulation, diversity-enhanced evolutionary exploration, and progress-dependent search-space adjustment to improve adaptability in noisy, non-convex, and non-stationary optimization environments.
- A sliding-window-based second-order momentum mechanism is developed together with crossover, mutation, elite preservation, and adaptive boundary adjustment. The integrated mechanisms jointly improve empirical optimization stability, preserve population diversity, and balance global exploration with local refinement.
- Comprehensive experiments are conducted on synthetic benchmark functions and practical tasks, including image classification, Transformer training, reinforcement learning, neural architecture search, and UAV path planning. The experiments evaluate the DSOF in terms of convergence stability, global-optimum discovery, solution diversity, search efficiency, and computational cost.

The remainder of the paper is organized as follows. Section 2 reviews related studies on evolutionary and metaheuristic optimization, adaptive gradient methods, restart strategies, and stochastic variance control. Section 3 presents the proposed DSOF framework including three main submodules (the DSMM, HMSDE, and APSA), together with the experimental protocol and evaluation results. Section 4 discusses the empirical findings, the complementary roles of the three mechanisms, and the limitations of the current study. Finally, Section 5 concludes the paper and outlines directions for future research.

2. Related work

The proposed DSOF draws on three research areas: evolutionary algorithms for global optimization, adaptive gradient methods for non-convex learning, and gradient variance control techniques in stochastic optimization.

2.1. Evolutionary and metaheuristic algorithms for global optimization search

Evolutionary algorithms, such as genetic algorithms (GAs), particle swarm optimization (PSO), and ant colony optimization (ACO), have been extensively studied for global optimization, particularly in combinatorial and multimodal landscapes. Sarkar and Biswas [12] introduced a score-based framework for comparing evolutionary algorithms according to multiple performance criteria. Ghimire et al. [13] applied parallel ACO to quantum-computing problems. Related developments include the extensions of swarm intelligence for constrained optimization [14], and opposition-based multi-objective Adam for combining gradient-based and population-based search [15].

Modern black-box optimizers provide more advanced global-search mechanisms. The covariance matrix adaptation evolution strategy (CMA-ES) adapts the mean, covariance matrix, and step size of a Gaussian search distribution [9], while the NES updates the distribution parameters using the natural gradient of expected fitness [10]. Bayesian optimization selects candidates through a surrogate model and an acquisition function and is particularly suitable for expensive, relatively low-dimensional objectives. In contrast, the DSOF retains stochastic-gradient-based local optimization and periodically applies crossover and mutation, thereby combining efficient local descent with population-based exploration.

2.2. Adaptive gradient methods for deep and non-convex optimization

Adaptive learning-rate methods are fundamental to efficient deep learning. Variants such as AdaLip [5] and AdaGrad with momentum [16] aim to balance stability and responsiveness to gradient dynamics. Reyad et al. [17] proposed a modified Adam algorithm to alleviate overfitting in deep networks, whereas Verma and Maiti [4] introduced statistical weighting into gradient updates. Other studies have examined the respective roles of adaptive learning rates and momentum [6]. Sun et al. [18] introduced AdaSAM, which combines adaptive learning rates and momentum within the sharpness-aware minimization framework. Levin et al. [19] analyzed how smooth parameterizations affect non-convex optimization landscapes.

Restart-based strategies provide another mechanism for renewing exploration. Stochastic gradient descent with warm restarts periodically increases the learning rate according to a cosine schedule [11]. Unlike such predefined restart schedules, the DSOF adjusts the learning rate using recent gradient-variance information and modifies the parameter search space according to objective improvement.

2.3. Variance reduction and stability in stochastic optimization

A central challenge in stochastic optimization is controlling gradient variance and improving convergence stability. References [2, 3] provide theoretical analyses of non-convex stochastic optimization in the presence of gradient uncertainty. The DSOF builds on the same motivation by introducing variance-aware step-size regulation and a dynamic noise-suppression mechanism. Islamov et al. [20] investigated how neural-network architecture affects loss-landscape geometry, showing that deeper and wider architectures may produce sharper curvature and stronger non-convexity.

In contrast to methods that focus primarily on either global exploration or local convergence, the DSOF combines adaptive step-size regulation, periodic evolutionary exploration, and dynamic search-space adjustment within a unified framework.

3. Algorithm framework

This section presents the DSOF, which comprises three core submodules: the DSMM, the hybrid metaheuristic strategy for diversity-enhanced exploration (HMSDE), and the APSA. The algorithmic procedure, experimental protocol, component-wise evaluation, practical validation, and baseline comparisons are subsequently presented.

3.1. Main steps and algorithm

To address the three core challenges in the proposed optimization problem, namely, gradient noise, premature convergence, and dynamic non-stationarity, three corresponding mechanisms are designed. The proposed DSOF consists of three principal steps.

Step 1. DSMM

Traditional adaptive gradient methods, such as Adam and RMSProp, estimate the first-order moment (mean) and second-order moment (uncentered variance) of gradients using an exponentially weighted moving average (EMA) [21]:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{3.1}$$

$$a_t = \beta_2 a_{t-1} + (1-\beta_2) g_t \odot g_t, \tag{3.2}$$

where g_t is the stochastic gradient at step t, $\beta_1, \beta_2 \in (0,1)$ are decay factors, and $\odot$ denotes the Hadamard product.

However, the exponential moving average inherently assigns larger weights to recent gradients, thereby amplifying the impact of noise, particularly in high-variance or dynamically changing objective functions. Specifically, when $\beta_2 = 0.999$, more than 63% of the cumulative weight is assigned to the most recent 100 iterations, as reported in [22]. Such weighting may increase sensitivity to recent gradient fluctuations and affect the accuracy of the estimated update direction. To address this issue,

the DSOF replaces the EMA-based variance estimator with a sliding-window second-order momentum mechanism, which calculates gradient statistics over a fixed-size historical window.

Let W be the window size. We define a buffer that stores the most recent W gradient vectors $\{g_{t-W+1}, \ldots, g_t\}$. Based on these vectors, the mean and variance of the recent gradients are estimated. The mean vector is computed as

$$\mu_t = \frac{1}{W} \sum_{i=t-W+1}^{t} g_i \in \mathbb{R}^d, \tag{3.3}$$

which assigns equal weights to all gradients within the window, thereby avoiding the tendency of the EMA to overemphasize the most recent noisy updates. To mitigate the $O(Wd)$ computational cost associated with directly recomputing the variance at each iteration, Welford's online algorithm is employed to incrementally update the windowed statistics. This implementation requires $O(d)$ time per iteration and maintains numerical stability for high-dimensional parameters. The sliding-window variance is computed element-wise as

$$v_t^{(\text{window})} = \frac{1}{W} \sum_{i=t-W+1}^{t} (g_i - \mu_t) \odot (g_i - \mu_t) \in \mathbb{R}^d. \tag{3.4}$$

Then, to balance local variance estimation and long-term stability, we integrate the sliding-window estimate with an exponential moving average:

$$v_t = \beta v_{t-1} + (1 - \beta) v_t^{(\text{window})}, \tag{3.5}$$

where $\beta \in (0, 1)$ is the smoothing parameter. Since both v_{t-1} and $v_t^{(\text{window})}$ belong to $\mathbb{R}^d$, the operations in Eq (3.5) are performed element-wise. The EMA smoothing term improves robustness against abrupt variance spikes. This hybrid design balances local responsiveness and long-term stability: the sliding window captures recent gradient variability, whereas the EMA smooths sudden fluctuations.

The learning rate is a crucial hyperparameter in optimization algorithms, as it determines the step size for iterative updates during gradient descent. Rather than using a fixed learning rate, the dynamic second-order momentum mechanism adjusts the learning rate according to gradient variance.

The variance vector is first scalarized as

$$\bar{v}_t = \frac{1}{d} \sum_{k=1}^{d} v_{t,k}. \tag{3.6}$$

The adaptive learning rate η_t is then computed as

$$\eta_t = \frac{\eta_0}{\sqrt{\bar{v}_t} + \epsilon} \cdot \frac{1}{1 + \lambda \max(0, \bar{v}_t - \sigma_{\max}^2)}, \tag{3.7}$$

where $\epsilon > 0$ is a small constant for numerical stability. This formulation ensures that the step size adapts to both local noise conditions captured by the sliding window and global stability trends captured by the EMA. Additionally, when high gradient variance occurs, an additional decay term is applied to reduce the learning rate and suppress unstable updates.

$$\bar{v}_t > \sigma_{\max}^2 \Longrightarrow \eta_t \leftarrow \eta_t \exp(-\lambda \bar{v}_t), \tag{3.8}$$

where $\sigma^2_{\max}$ denotes the predefined noise-tolerance threshold, set to 1.0 in the experiments, and λ is the exponential suppression coefficient. The resulting learning rate is clipped to the interval $[\eta_{\min}, \eta_{\max}]$ before updating the parameters.

In summary, the DSMM mechanism replaces the exponentially decaying weights of the EMA with uniformly weighted historical gradients. By uniformly aggregating historical gradients within a fixed-size window, the proposed estimator reduces the excessive influence of individual recent gradients under noisy conditions. Furthermore, variance-based step-size scaling and noise suppression dynamically regulate update magnitudes according to gradient uncertainty, thereby reducing oscillations and decreasing the risk of unstable updates in evolving optimization processes. The sliding-window-based variance estimation and adaptive step-size adjustment enhance robustness to gradient noise and improve the adaptability of the DSOF in non-stationary environments.

Step 2. HMSDE

Traditional gradient-based optimization algorithms are susceptible to premature convergence, particularly in multimodal objective landscapes. This limitation arises from reliance on local information and insufficient global exploration. To address this issue, the DSOF introduces a hybrid metaheuristic strategy that periodically incorporates global search to enhance population diversity and explore new regions of the solution space.

At the t-th iteration, the DSOF maintains a population buffer $\mathcal{P}_t$, which stores a fixed-size set of candidate parameter vectors and the corresponding fitness scores. The buffer provides a diverse pool of high-quality candidate solutions for crossover and mutation, and preserves an elite archive of high-quality solutions across iterations. The population buffer is refreshed every K iterations using newly generated candidates and performance metrics.

Every K iterations, the DSOF selects candidate parameter vectors θ_i from $\mathcal{P}_t$ according to their fitness values. For each selected parent, crossover or mutation is used to generate an offspring candidate. When crossover is selected, another parent θ_j is randomly chosen from $\mathcal{P}_t$, and a new candidate is generated through linear interpolation:

$$\theta' = \alpha\theta_i + (1-\alpha)\theta_j, \qquad \alpha \sim U(0,1). \tag{3.9}$$

This operation combines information from two well-performing individuals. The interpolation coefficient α is sampled from a uniform distribution, enabling symmetric exploration between the parents. This strategy preserves strong candidate information while exploring intermediate regions.

When mutation is selected, a Gaussian perturbation is applied to the selected candidate with probability p_m:

$$\theta' = \theta_i + \theta_{Gauss}, \qquad \theta_{Gauss} \sim \mathcal{N}(0, \sigma_t^2 I_d). \tag{3.10}$$

The mutation strength σ_t decays exponentially over time:

$$\sigma_t = \sigma_0 \exp(-t/\tau), \tag{3.11}$$

where σ_0 is the initial mutation strength and τ controls the decay rate. This schedule supports broad exploration during the early stages and fine-grained refinement during the later stages.

To quantify exploration efficiency, population diversity is defined as the average Euclidean distance from the population mean:

$$\bar{\theta} = \frac{1}{M}\sum_{i=1}^{M}\theta_i, \tag{3.12}$$

$$\text{Diversity} = \frac{1}{M}\sum_{i=1}^{M}\|\theta_i - \bar{\theta}\|_2, \tag{3.13}$$

where M denotes the population size. The diversity metric is monitored to determine whether the population converges toward a narrow region of the search space.

The hybrid metaheuristic strategy addresses the challenge posed by multiple local optima by combining periodic crossover, adaptive mutation, and elitism. The combined operations preserve population diversity, provide heuristic global exploration, and complement the local optimization capability of gradient-based updates.

Step 3. APSA

To balance exploration and exploitation, the DSOF incorporates an autonomous parameter-space adjustment mechanism that modifies the search boundaries according to optimization progress.

Let

$$f_t^* = \min_{0\le s\le t} f(\theta_s) \tag{3.14}$$

denote the best objective value observed up to iteration t. The value $f^*_{t-T_{\text{det}}}$ denotes the best objective value observed T_{det} iterations earlier. The average improvement over the detection interval is defined as

$$\Delta f = \frac{|f_t^* - f^*_{t-T_{\text{det}}}|}{T_{\text{det}}}. \tag{3.15}$$

If $\Delta f \le \epsilon_{\text{det}}$, optimization stagnation is detected. Let $R^{(k)}$ denote the current range of the k-th coordinate:

$$R^{(k)} = \theta^{(k)}_{\max} - \theta^{(k)}_{\min}, \qquad k = 1, \ldots, d. \tag{3.16}$$

When stagnation is detected, the bounds are expanded element-wise:

$$\left[\theta^{(k),\text{new}}_{\min}, \theta^{(k),\text{new}}_{\max}\right] \leftarrow \left[\theta^{(k)}_{\min} - \delta R^{(k)}, \theta^{(k)}_{\max} + \delta R^{(k)}\right]. \tag{3.17}$$

When $\Delta f > \gamma$, indicating sufficient progress, the bounds are contracted:

$$\left[\theta^{(k),\text{new}}_{\min}, \theta^{(k),\text{new}}_{\max}\right] \leftarrow \left[\theta^{(k)}_{\min} + \frac{\kappa R^{(k)}}{2}, \theta^{(k)}_{\max} - \frac{\kappa R^{(k)}}{2}\right], \tag{3.18}$$

where $\delta, \kappa \in (0, 1)$ denote the expansion and contraction ratios, respectively. After each adjustment, the parameter vector and population candidates are projected onto the updated hyper-rectangle $\prod_{k=1}^{d}\left[\theta^{(k),\text{new}}_{\min}, \theta^{(k),\text{new}}_{\max}\right]$ to ensure feasibility.

The APSA expands the search range when optimization stagnates and contracts the range when sufficient improvement is observed. The resulting search-space regulation coordinates broader exploration with local refinement.

Based on the three mechanisms, the complete DSOF procedure is presented in Algorithm 1.

Algorithm 1 DSOF

Input: Parameter vector $\theta \in \mathbb{R}^d$ (d: parameter dimension), initial learning rate η_0, gradient window size W, momentum decay factor β, numerical stability constant ϵ, penalty coefficient λ, metaheuristic interval K, initial mutation strength σ_0, mutation decay coefficient τ, boundary expansion ratio δ, boundary contraction ratio κ, stagnation threshold γ, population size M, stagnation detection window T_{det}, stagnation criteria ϵ_{det}, min(max) learning rate $\eta_{\min}(\eta_{\max})$, max iterations $T_{\max}$, loss threshold f_{thresh}, elite injection rate ρ.

Output: Optimized parameter θ^*, loss history $\mathcal{F} = \{f_1, f_2, \ldots, f_T\}$ (f_t means the loss function value at step t).

Step 0: Initialization

$t \leftarrow 0, \mathcal{B}_0 \leftarrow \emptyset, v_{-1} \leftarrow 0, \mathcal{P}_0 \leftarrow \{\theta_0\}, f_0^* \leftarrow +\infty, \mathcal{F}^* \leftarrow \{f_0^*\}, \theta_{\min} \leftarrow -5 \cdot \mathbf{1}_d, \theta_{\max} \leftarrow 5 \cdot \mathbf{1}_d, \mathcal{F} \leftarrow \emptyset$.

Step 1: Main optimization loop

While $t < T_{\max}$ and $f_t^* > f_{\text{thresh}}$ **do**

Step 1.1: Gradient computation

Compute current gradient: $g_t \leftarrow \nabla f(\theta_t)$; compute current loss: $f_{\text{current}} \leftarrow f(\theta_{t-1})$;

Increment iteration: $t \leftarrow t + 1$;

Step 1.2: DSMM

Step 1.2.1: Gradient history window maintenance

Append current gradient: $\mathcal{B}_t \leftarrow \mathcal{B}_{t-1} \cup \{g_t\}$;

If $|\mathcal{B}_t| > W$ **then**

  Remove oldest gradient: $\mathcal{B}_t \leftarrow \mathcal{B}_t \setminus \{g_{t-W}\}$;

End If

Step 1.2.2: Window gradient statistics computation

Form the gradient matrix:

  $G_t \leftarrow [g_{t-W+1}, g_{t-W+2}, \ldots, g_t] \in \mathbb{R}^{W \times d}$;

Compute mean: $\mu_t \leftarrow \frac{1}{W} \sum_{i=t-W+1}^{t} g_i$;

Compute variance: $v_{(\text{win}),t} \leftarrow \frac{1}{W} \sum_{i=t-W+1}^{t} (g_i - \mu_t) \odot (g_i - \mu_t)$;

Step 1.2.3: EMA-smoothed variance estimation

Update variance: $v_t \leftarrow \beta \cdot v_{t-1} + (1 - \beta) \cdot v_{(\text{win}),t}$;

Store for next iteration: $v_{t-1} \leftarrow v_t$;

Step 1.2.4: Adaptive learning rate calculation

Compute scalarized variance: $\bar{v}_t \leftarrow \frac{1}{d} \sum_{k=1}^{d} v_{t,k}$;

Calculate base learning rate: $\eta_t \leftarrow \frac{\eta_0}{\sqrt{\bar{v}_t} + \epsilon}$;

Step 1.2.5: Variance-based penalty

If $\bar{v}_t > 1.0$ **then**

  Apply penalty: $\eta_t \leftarrow \eta_t \cdot \exp(-\lambda \bar{v}_t)$;

End If

Stability adjustment: $\eta_t \leftarrow \frac{\eta_t}{1 + \lambda \cdot \max(0, \bar{v}_t - 1.0)}$;

Step 1.2.6: Learning rate clipping

Clip to valid range: $\eta_t \leftarrow \max(\eta_{\min}, \min(\eta_t, \eta_{\max}))$;

Step 1.3: HMSDE

If $t \bmod K == 0$ **then (Execute every K iterations)**

Continued on next page

Algorithm 1 DSOF

Step 1.3.1: Population retrieval

$\mathcal{P}_{\text{curr}} \leftarrow \mathcal{P}_{t-1}$; $\mathcal{O} \leftarrow \emptyset$;

Step 1.3.2: Offspring generation

For Each $\theta_i \in \mathcal{P}_{\text{curr}}$ **do**

  If rand$(0, 1) < 0.5$ **then (Crossover)**

    Select random parent: $\theta_j \leftarrow \text{randselect}(\mathcal{P}_{\text{curr}} \setminus \{\theta_i\})$;

    Sample coefficient: $\alpha \sim U(0, 1)$;

    Linear crossover: $\theta' \leftarrow \alpha \cdot \theta_i + (1 - \alpha) \cdot \theta_j$;

  Else (Mutation)

    Compute decayed mutation strength: $\sigma_t \leftarrow \sigma_0 \cdot \exp\left(-\frac{t}{\tau}\right)$;

    Gaussian mutation: $\theta' \leftarrow \theta_i + \theta_{Gauss}, \quad \theta_{Gauss} \sim \mathcal{N}(0, \sigma_t^2 I_d)$;

  End If

  Add to offspring: $\mathcal{O} \leftarrow \mathcal{O} \cup \{\theta'\}$;

End For

End If

Step 1.3.3: Elite selection

Combine parents and offspring: $\mathcal{C} \leftarrow \mathcal{P}_{\text{curr}} \cup \mathcal{O}$;

Sort by fitness: $\mathcal{C} \leftarrow \text{sort}(\mathcal{C}, \text{key} = f(\cdot))$;

Determine elite count: $E \leftarrow \max(1, \lfloor 0.1 \cdot |\mathcal{C}| \rfloor)$;

Select elites: $\mathcal{E} \leftarrow \{C_1, C_2, \ldots, C_E\}$;

Step 1.3.4: Elite injection

Best candidate from population: $\theta_{\text{best}} \leftarrow C_1$

Evaluate the best candidate: $f_{\text{best}} \leftarrow f(\theta_{\text{best}})$

Initialize the temporary parameter: $\theta_{\text{temp}} \leftarrow \theta_{t-1}$

Initialize the temporary loss: $f_{\text{temp}} \leftarrow f_{\text{current}}$

If $f_{\text{best}} < f_{\text{current}}$ **then**

  Elite-injected candidate: $\theta_{\text{temp}} \leftarrow (1 - \rho)\theta_{t-1} + \rho\theta_{\text{best}}$;

  Project onto current bounds: $\theta_{\text{temp}} \leftarrow \max(\theta_{\min}, \min(\theta_{\text{temp}}, \theta_{\max}))$;

  Temporary loss: $f_{\text{temp}} \leftarrow f(\theta_{\text{temp}})$;

  If $f_{\text{temp}} < f_{\text{current}}$ **then**

    Accept candidate: $\theta_{t-1} \leftarrow \theta_{\text{temp}}$;

    Update current loss: $f_{\text{current}} \leftarrow f_{\text{temp}}$;

  End If

End If

Step 1.3.5: Population update

Update population: $\mathcal{P}_t \leftarrow \mathcal{E} \cup \{C_{E+1}, \ldots, C_M\}$;

Step 1.4: APSA

Step 1.4.1: Best loss update

Update best loss: $f_t^* \leftarrow \min(f_{t-1}^*, f_{\text{current}})$;

Define the corresponding optimal parameter vector:

  $\theta_t^* = \arg\min_{k \le t} f(\theta_k)$

Store in history: $\mathcal{F}^* \leftarrow \mathcal{F}^* \cup \{f_t^*\}$;

Continued on next page

Algorithm 1 DSOF

Step 1.4.2: Stagnation detection

Initialize improvement: $\Delta f \leftarrow 1.0$;

If $|\mathcal{F}^*| > T_{\text{det}}$ **then**

  Current best: $f_t^* \leftarrow \mathcal{F}^*[-1]$;

  Past best: $f_{t-T_{\text{det}}}^* \leftarrow \mathcal{F}^*[-(T_{\text{det}}+1)]$;

  Improvement rate: $\Delta f \leftarrow \frac{|f_t^* - f_{t-T_{\text{det}}}^*|}{T_{\text{det}}}$;

End If

Step 1.4.3: Boundary adjustment

Compute current range: $R \leftarrow \theta_{\max} - \theta_{\min}$;

If $\Delta f \leq \epsilon_{\text{det}}$ **then (Stagnation detected)**

  Expand bounds: $\theta_{\min} \leftarrow \theta_{\min} - \delta \cdot R$;

  $\theta_{\max} \leftarrow \theta_{\max} + \delta \cdot R$;

ElsIf $\Delta f > \gamma$ **then (Good progress)**

  Contract bounds: $\theta_{\min} \leftarrow \theta_{\min} + \frac{\kappa \cdot R}{2}$;

  $\theta_{\max} \leftarrow \theta_{\max} - \frac{\kappa \cdot R}{2}$;

End If

Step 1.4.4: Parameter constraint

Clip to bounds: $\theta_{t-1} \leftarrow \max(\theta_{\min}, \min(\theta_t^*, \theta_{\max}))$;

Step 1.5: Parameter update

Gradient descent update: $\theta_t \leftarrow \theta_{t-1} - \eta_t \cdot g_t$;

Record loss: $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_t\}$;

End While

Step 2: Finalization

Return results: $\theta^* \leftarrow \theta_t$;

return $\theta^*, \mathcal{F}$.

3.2. Illustrative execution and ablation settings of the DSOF

To illustrate the execution of Algorithm 1 and clarify the ablation settings used in the experiments, consider a two-dimensional optimization problem at iteration $t \pmod K = 0$ with the current parameter vector $\theta_t = (1.0, 0.8)$, a population containing several candidate solutions, and a metaheuristic update activated. The DSMM component first computes the recent gradient mean and variance from the sliding window and obtains the gradient-based candidate

$$\theta_t^{(g)} = \theta_{t-1} - \eta_t g_t. \tag{3.19}$$

The HMSDE component then generates additional candidates from the current population, after which elite selection and elite injection determine whether a population candidate should be incorporated into the current solution. Finally, the APSA expands or contracts the parameter bounds according to the recent improvement rate.

For the DSOF (Full), each selected parent may generate an offspring through either crossover or mutation. For example, given two parent vectors $\theta_i = (0.9, 0.7)$ and $\theta_j = (1.1, 0.6)$, and $\alpha = 0.4$,

crossover produces

$$\theta' = 0.4\theta_i + 0.6\theta_j = (1.02, 0.64). \tag{3.20}$$

Alternatively, mutation produces

$$\theta' = \theta_i + \theta_{Gauss}, \quad \theta_{Gauss} \sim \mathcal{N}(0, \sigma_t^2 I_d), \tag{3.21}$$

which introduces a local random perturbation. The parent and offspring populations are then combined, ranked according to the objective value, and used for elite selection and injection.

DSOF (No Mutate) follows the same procedure as DSOF (Full), except that the mutation branch is disabled. Therefore, all offspring generated during the HMSDE stage are obtained through crossover:

$$\theta' = \alpha\theta_i + (1 - \alpha)\theta_j. \tag{3.22}$$

This variant is used to isolate the contribution of mutation to population diversity and local refinement.

DSOF (No Cross) disables the crossover branch and generates offspring only through Gaussian mutation:

$$\theta' = \theta_i + \theta_{Gauss}, \quad \theta_{Gauss} \sim \mathcal{N}(0, \sigma_t^2 I_d). \tag{3.23}$$

This variant is used to evaluate the contribution of crossover to information exchange among candidate solutions and the effect on global exploration. All other components, including the DSMM, elite selection, elite injection, APSA, stopping criteria, and hyperparameter settings, remain unchanged across the three variants.

Accordingly, the three experimental variants differ only in the offspring-generation operation of Step 1.3.2 in Algorithm 1. DSOF (Full) uses both crossover and mutation, DSOF (No Mutate) uses crossover only, and DSOF (No Cross) uses mutation only. The controlled design enables the individual contributions of the two evolutionary operators to be evaluated in Section 3.4.

3.3. Computational complexity analysis

Let d denote the parameter dimension, W denote the gradient-window size, M denote the population size, K denote the metaheuristic activation interval, and C_f and C_g denote the costs of one objective function evaluation and one stochastic gradient evaluation, respectively.

For the DSMM, stochastic-gradient computation requires $O(C_g)$ time. Maintaining the gradient-history window requires $O(d)$ time per iteration when incremental statistics are adopted. The windowed mean and variance can also be updated in $O(d)$ time using running statistics. Direct recomputation from all W stored gradients instead requires a complexity of $O(Wd)$. Variance smoothing, adaptive learning-rate calculation, parameter updating, and projection each require $O(d)$ time. Therefore, the per-iteration time complexity of the gradient-based stage is

$$O(C_g + d), \tag{3.24}$$

with incremental window statistics, or

$$O(C_g + Wd), \tag{3.25}$$

with direct recomputation.

The HMSDE is activated once every K iterations. Generating M offspring through crossover or mutation requires $O(Md)$ time. Evaluating the parent and offspring populations requires $O(MC_f)$ time, while sorting at most $2M$ candidates requires $O(M \log M)$ time. Elite selection and elite injection require an additional $O(M + d)$ time. Therefore, the time complexity of one HMSDE activation is

$$O(Md + MC_f + M \log M). \tag{3.26}$$

The amortized per-iteration contribution of the HMSDE is

$$O\left(\frac{Md + MC_f + M \log M}{K}\right). \tag{3.27}$$

The APSA computes the historical improvement rate, updates the parameter bounds, and projects the current solution onto the adjusted domain. The corresponding operations require $O(d)$ time per activation. Consequently, the amortized time complexity of one DSOF iteration is

$$O\left(C_g + d + \frac{Md + MC_f + M \log M}{K}\right), \tag{3.28}$$

when incremental sliding-window statistics are adopted. Under direct recomputation of the window statistics, the corresponding complexity becomes

$$O\left(C_g + Wd + \frac{Md + MC_f + M \log M}{K}\right). \tag{3.29}$$

Over $T_{\max}$ iterations, the total time complexity is

$$O\left[T_{\max}\left(C_g + d + \frac{Md + MC_f + M \log M}{K}\right)\right], \tag{3.30}$$

with incremental window statistics, or

$$O\left[T_{\max}\left(C_g + Wd + \frac{Md + MC_f + M \log M}{K}\right)\right], \tag{3.31}$$

with direct window-statistic recomputation.

For space complexity, the current parameter vector, gradient, moment statistics, and adaptive bounds require $O(d)$ memory. The sliding gradient window stores at most W gradient vectors and requires $O(Wd)$ memory. The population and offspring buffers require $O(Md)$ memory, while the fitness values and complete loss history require $O(M + T_{\max})$ memory. Therefore, the overall space complexity is

$$O(Wd + Md + T_{\max}). \tag{3.32}$$

When only a fixed-length loss history is retained, the space complexity reduces to

$$O((W + M)d). \tag{3.33}$$

3.4. Hyperparameter sensitivity analysis

DSOF hyperparameters can be divided into three categories. Gradient-related parameters, including W, β, η_0, and λ, control noise suppression and update stability. Evolutionary parameters, including M, K, p_m, and σ_0, regulate global exploration and computational cost. Boundary-adjustment parameters, including δ, κ, T_{det}, and ϵ_{det}, determine the sensitivity of search-space expansion and contraction. The complete hyperparameter configurations for the DSOF and the baseline optimizers are summarized in Table 1.

Table 1. Hyperparameters and training settings for the DSOF and the baseline optimizers.

Setting	**Value**
Initial Parameter Vector θ_0	Task-dependent random initialization
Parameter Dimension d	Determined by the corresponding task
Population Size M	20
Trials / Seeds	10 trials, seeds 1–10
Statistical Test	Two-sided paired t-test, exact p-values reported
Learning Rate (DSOF)	$\eta_t = \eta_0/\sqrt{t},\ \eta_0 = 0.001$
Window Size W	50
Momentum Decay Factor β	0.97
Exploration Interval K	25
Mutation Probability p_m	0.2
Initial Mutation Strength σ_0	0.05
Mutation Decay Parameter τ	400
Penalty Coefficient λ	0.05
Variance Threshold $\sigma^2_{\max}$	1.0
Expansion Ratio δ	0.02
Scaling Factors	$\kappa = 0.05$, $\epsilon = 10^{-8}$, $\gamma = 0.01$
Stagnation Detection Window T_{det}	50
Stagnation Criterion ϵ_{det}	10^{-5}
Minimum Learning Rate $\eta_{\min}$	10^{-6}
Maximum Learning Rate $\eta_{\max}$	0.05
Maximum Number of Iterations $T_{\max}$	3000
Loss Threshold f_{thresh}	Task-dependent
Elite Injection Rate ρ	0.3
Initial Lower Bound $\theta_{\min}$	$-5 * \mathbf{1}_d$
Initial Upper Bound $\theta_{\max}$	$5 * \mathbf{1}_d$
Batch Size (Adam/AdamW)	128, lr = 10^{-3}
Batch Size (RMSProp)	128, lr = 10^{-3}, decay = 0.99
Batch Size (SGD-M)	128, momentum = 0.9
Early Stopping	20 epochs without improvement
Hardware	NVIDIA A100 GPU, 40 GB, 48-hour cap per run
RL Environment	HalfCheetah, mujoco-py v2.1, horizon = 1000, reward scale = 0.1

Generally, larger values of W and β are suitable for noisy objectives, whereas larger values of M, smaller values of K, or larger mutation strengths may benefit highly multimodal problems. To evaluate the influence of the main DSOF hyperparameters, a one-factor-at-a-time sensitivity analysis is conducted for each selected parameter, while all remaining settings are fixed at the default values. The evaluated values are $W \in \{10, 25, 50, 100, 200\}$, $K \in \{10, 25, 50, 100\}$, $p_m \in \{0.05, 0.1, 0.2, 0.3, 0.5\}$, $\sigma_0 \in \{0.01, 0.03, 0.05, 0.1, 0.2\}$, $\beta \in \{0.90, 0.95, 0.97, 0.99\}$. All configurations are evaluated on the 50-dimensional Ackley function with Gaussian gradient noise of standard deviation 0.01. Each configuration is evaluated over 10 independent runs using the same set of random seeds and stopping criteria. The maximum number of iterations is set to 1200. The evaluation metrics include the best objective value achieved during optimization and the average runtime. The best final objective value is defined as

$$f_{\text{best}} = \min_{0 \le t \le T_{\max}} f(\theta_t). \tag{3.34}$$

The results are reported as the mean and standard deviation over 10 independent runs.

Table 2. Hyperparameter sensitivity analysis of the DSOF on the 50-dimensional Ackley function. Results are reported as mean ± standard deviation over 10 independent runs.

Parameter	Value	Best Final Objective Value	Runtime (s)
W	10	7.616711 ± 0.349328	**0.0853 ± 0.0022**
W	25	**7.616706 ± 0.349326**	0.0914 ± 0.0015
W	50	7.616733 ± 0.349353	0.1059 ± 0.0013
W	100	7.616855 ± 0.349256	0.1252 ± 0.0015
W	200	7.618490 ± 0.349068	0.1778 ± 0.0107
K	10	**7.614560 ± 0.350172**	0.1305 ± 0.0029
K	25	7.616733 ± 0.349353	0.1024 ± 0.0015
K	50	7.616844 ± 0.349252	0.0934 ± 0.0003
K	100	7.617124 ± 0.349362	**0.0909 ± 0.0016**
p_m	0.05	7.616726 ± 0.349339	0.1068 ± 0.0014
p_m	0.10	7.616733 ± 0.349337	0.1048 ± 0.0019
p_m	0.20	7.616733 ± 0.349353	**0.1029 ± 0.0013**
p_m	0.30	7.618806 ± 0.349274	0.1050 ± 0.0027
p_m	0.50	**7.609867 ± 0.358431**	0.1036 ± 0.0026
σ_0	0.01	7.616718 ± 0.349339	**0.1022 ± 0.0009**
σ_0	0.03	**7.611004 ± 0.356837**	**0.1022 ± 0.0009**
σ_0	0.05	7.616733 ± 0.349353	0.1042 ± 0.0010
σ_0	0.10	7.616734 ± 0.349353	0.1054 ± 0.0015
σ_0	0.20	7.616730 ± 0.349348	0.1032 ± 0.0012
β	0.90	7.616731 ± 0.349326	0.1040 ± 0.0020
β	0.95	7.616723 ± 0.349340	**0.1027 ± 0.0014**
β	0.97	7.616733 ± 0.349353	0.1034 ± 0.0016
β	0.99	**7.616709 ± 0.349326**	0.1062 ± 0.0014

The results in Table 2 show that the DSOF remains stable over a reasonably broad range of

hyperparameter values. A small gradient-window size responds rapidly to gradient changes but produces noisier variance estimates, whereas an excessively large window may delay adaptation to non-stationary gradients. Moderate values of W, generally between 25 and 100, provide a favorable balance between noise suppression and responsiveness. A small K strengthens evolutionary exploration but increases computational overhead, whereas a large K reduces the influence of crossover and mutation. Values between 25 and 50 generally provide stable performance across the evaluated tasks.

The mutation-related parameters mainly affect the exploration–exploitation trade-off. Large values of p_m or σ_0 increase population diversity but may introduce unstable perturbations, whereas very small values reduce the ability to escape local optima. Under the evaluated settings, $p_m \in [0.1, 0.3]$ and $\sigma_0 \in [0.03, 0.1]$ provide robust performance. A relatively large smoothing factor suppresses abrupt variance fluctuations, with $\beta \in [0.95, 0.99]$ producing stable results. For the APSA, moderate expansion and contraction ratios, generally between 0.02 and 0.05, avoid excessive boundary changes and insufficient adaptation.

Overall, the sensitivity results indicate that the DSOF is relatively robust to the evaluated hyperparameter settings. The default configuration $W = 50$, $K = 25$, $p_m = 0.20$, $\sigma_0 = 0.05$, and $\beta = 0.97$ does not always achieve the numerically lowest objective value under each individual parameter setting but provides a balanced configuration in terms of optimization performance, exploration frequency, stability, and runtime. The default values should therefore be regarded as practical initial settings rather than universally optimal choices.

In the following subsections, to ensure the reproducibility and statistical reliability, all experiments were repeated over 10 independent trials with random seeds 1–10. The mean and standard deviation across all trials are reported, and statistical significance was assessed using two-sided paired t-tests with exact p-values. Early stopping was applied when the validation performance did not improve for 20 consecutive epochs. All experiments were conducted on NVIDIA A100 GPUs with 40 GB of memory and a maximum runtime of 48 hours per run. The reinforcement learning environments followed the OpenAI Gym specifications [23]. For HalfCheetah [24], mujoco-py v2.1 was adopted with a horizon length of 1000 and a reward-scaling factor of 0.1.

3.5. Evaluation of the contribution of each component to optimization stability

This subsection analyzes the internal behavior and individual contributions of the three principal mechanisms in the DSOF. Unlike the practical-task evaluation in the next subsection, which compares the complete DSOF framework with external baselines, the experiments here focus on explaining how each mechanism affects variance estimation, population exploration, and parameter-space adaptation.

We first evaluate the contribution of the DSMM to reducing variance fluctuations and improving update stability.

Figure 1 illustrates the temporal behavior of the gradient-variance estimator during MNIST training. By assigning equal weights to all gradients within the sliding window, the DSMM reduces the disproportionate influence of individual recent gradients and produces a smoother estimate of gradient uncertainty. This experiment is intended to illustrate the internal variance-estimation process rather than to provide the final comparison with Adam and RMSProp, which is reported in Section 3.6.

The variance estimate v_t quantifies uncertainty in the parameter update direction. A larger variance

indicates larger variability among recent gradients and leads to a smaller adaptive learning rate, approximately following $\eta_t \propto 1/(\sqrt{\bar{v}_t} + \epsilon)$, thereby suppressing oscillatory updates. Figure 2 further shows the effect of the DSMM on the optimization trajectory under Gaussian gradient noise $\mathcal{N}(0, 0.1)$. The purpose of this experiment is to visualize how the sliding-window variance estimate suppresses abrupt loss oscillations and stabilizes the update process on the Rosenbrock function. The quantitative success-rate comparison among the DSOF, Adam, and RMSProp is reported separately in Section 3.6 to avoid duplicating the corresponding performance evaluation. Overall, Figures 1 and 2 show that the DSMM primarily contributes by smoothing the variance estimate and adapting the update magnitude according to directional uncertainty. These observations provide a mechanism-level explanation for the convergence stability and gradient-variance results subsequently reported in Section 3.6.

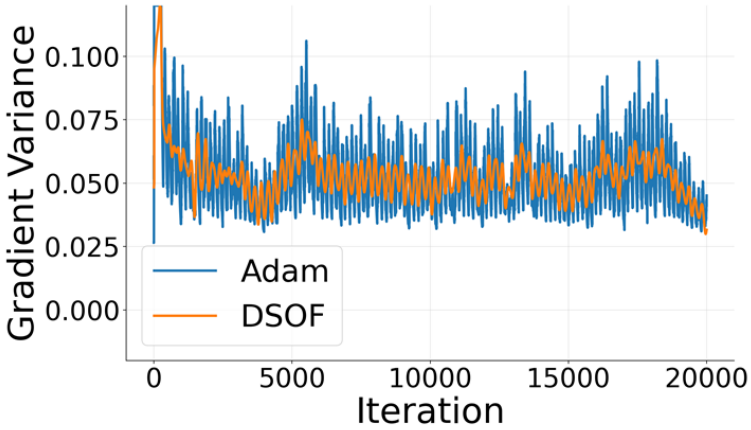


Figure 1. Gradient-variance estimation on MNIST.

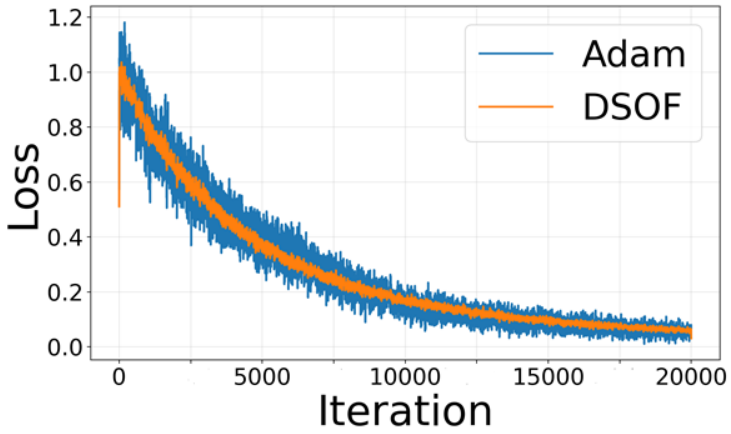


Figure 2. Training loss on the Rosenbrock function under Gaussian noise.

Second, we evaluate the contribution of the HMSDE to enhancing exploration capability and mitigate premature convergence caused by search-space limitations and local optima.

In the HMSDE, the top 10% of historical solutions are preserved after each evolutionary round to prevent stochastic updates from discarding high-quality candidates. Figure 3 shows the transformer training-loss trajectory when periodic crossover, mutation, and elite retention are interleaved with

gradient-based updates: between successive evolutionary rounds, the gradient update performs local refinement, whereas the periodic hybrid operations reintroduce alternative candidates outside the current narrow region, so that the search does not collapse onto a single local basin. The panel therefore functions as a process-level record of how the HMSDE alternates local refinement with population-based exploration. The separate effects of crossover and mutation (namely, faster convergence versus broader diversity preservation) are isolated in the DSOF (No Cross) and DSOF (No Mutate) ablations reported in Section 3.6, so they are not re-quantified here.

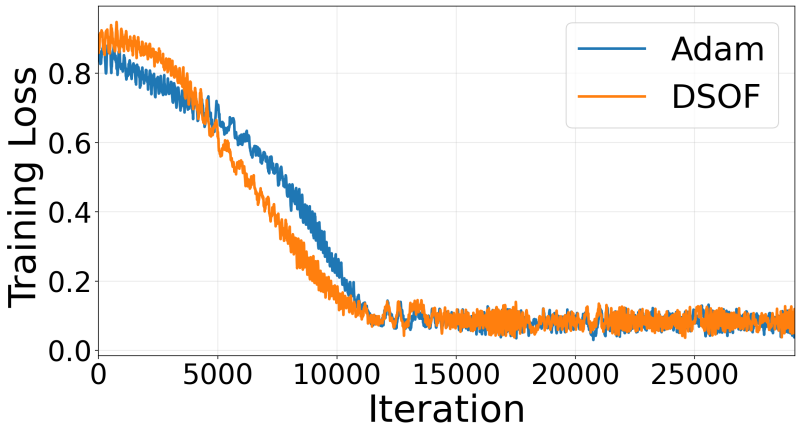


Figure 3. Transformer training loss on WikiText-2.

To improve adaptability while maintaining optimization stability, the APSA adjusts the parameter search space according to observed optimization progress. So, we evaluate the contribution of the APSA to adaptive search behavior in a UAV path-planning task.

To avoid instability caused by frequent boundary changes, a minimum adjustment interval of $\Delta_t = 20$ iterations is imposed for boundary updates, and Δf is smoothed using a moving average. The experiment considers a self-constructed three-dimensional UAV path-planning scenario within a bounded workspace containing randomly generated dynamic obstacles. The UAV is modeled as a point-mass system with a limited step size and collision constraints. A trial is considered successful when the UAV reaches the target region without collision within a fixed planning horizon. As shown in Figure 4, the APSA dynamically changes the admissible search radius according to the observed optimization progress. When the objective improvement becomes insufficient, the search range is expanded to increase exploration. When a stable improvement is observed, the range is contracted to concentrate the search around promising regions. Thus, Figure 4 is used to visualize the internal boundary-adjustment trajectory rather than to repeat the final path-length and obstacle-avoidance comparisons reported in Section 3.6.

Overall, the experiments in this subsection provide mechanistic evidence for the three components of the DSOF. The DSMM stabilizes variance estimation, the HMSDE alternates local refinement with population-based exploration, and the APSA adjusts the search range in response to optimization progress. The following subsection evaluates whether these mechanisms lead to improved task-level performance relative to external baseline algorithms.

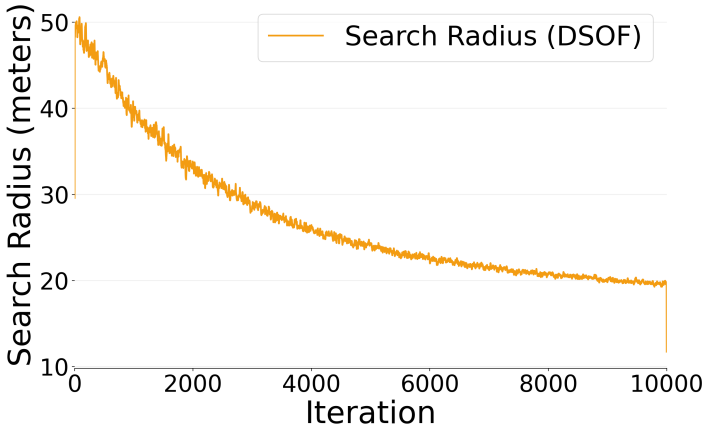


Figure 4. UAV path-planning results with adaptive search-space adjustment.

3.6. Evaluation on practical problems

This subsection evaluates the complete DSOF framework against representative baseline algorithms on synthetic optimization, image classification, reinforcement learning, transformer training, UAV path planning, and hyperparameter optimization. In contrast to Section 3.5, the emphasis here is on final task-level performance rather than the internal behavior of individual mechanisms.

3.6.1. Verification of the dynamic second-order momentum mechanism

In the noise robustness test, the DSOF is compared with Adam and RMSProp across three representative scenarios. On the Rosenbrock function [25], Gaussian noise sampled from $\mathcal{N}(0, 0.1)$ is injected into the gradient evaluations, and a run is declared successful when $f(x) < 10^{-4}$. For the MNIST classification task [26], a ResNet-18 model is trained and the fluctuation coefficient $\mathrm{Var}(v_t)/\,\mathrm{Mean}(v_t)$ of the gradient-variance estimator is recorded over training to assess noise suppression. In the reinforcement learning setting, the policy-gradient variance, the smoothness of the training curve, and the episodic cumulative reward are evaluated on the MuJoCo HalfCheetah environment. The cumulative-reward convergence curves are additionally reported to provide a direct comparison of learned-policy quality. Together, these three scenarios examine the DSMM from the perspectives of synthetic non-convex stability, empirical gradient-variance behavior in deep learning, and policy-optimization robustness under stochastic updates.

The results are shown in Table 3. To provide a more direct evaluation of reinforcement learning performance, the episodic cumulative-reward convergence curves are additionally reported for the HalfCheetah task. Each curve represents the mean cumulative reward across multiple independent runs, while the shaded region denotes the corresponding standard deviation. Episodic cumulative reward directly reflects the quality of the learned policy, whereas policy-gradient variance characterizes the stability of parameter updates.

As shown in Figure 5(a), Adam and the DSOF both achieve a success rate of 100% on the noisy Rosenbrock function, with a 95% Wilson confidence interval of 94.0%–100.0%. RMSProp achieves a success rate of 98.3%, with a corresponding confidence interval of 91.1%–99.7%. These results demonstrate that the DSOF maintains stable convergence performance under the evaluated noisy conditions, although it does not exhibit a statistically significant advantage over Adam with respect to

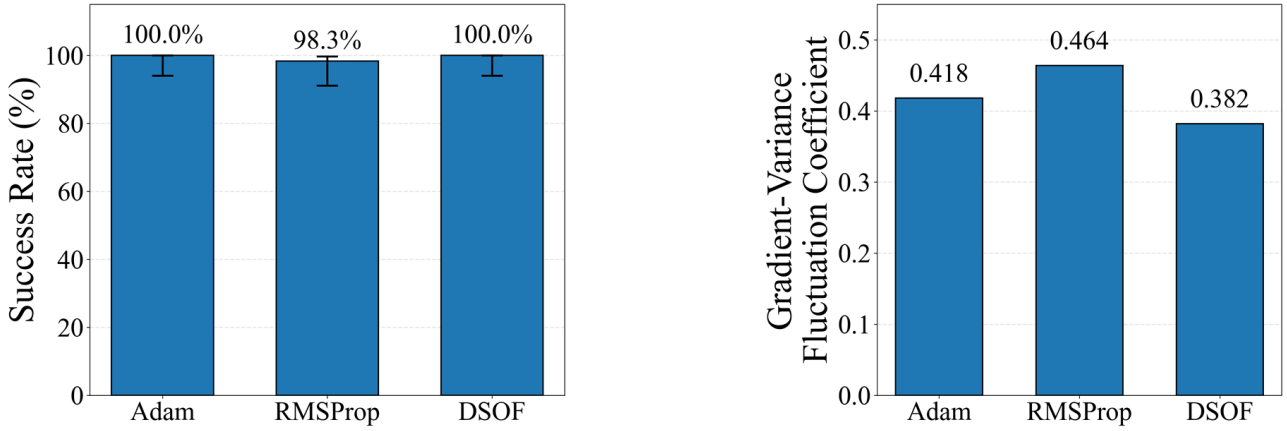
the success rate.

In the MNIST experiment, the DSOF with $W = 50$ reduces the gradient-variance coefficient to 0.382, which is 8.6% lower than that of Adam (0.418) and 17.7% lower than that of RMSProp (0.464), suggesting more stable gradient-variance behavior, as shown in Figure 5(b).

Together with the HalfCheetah reward results, these experiments indicate that the DSMM contributes to gradient stability while preserving competitive task-level performance.

Table 3. Verification results of the dynamic second-order momentum mechanism. Rosenbrock success rates are reported with 95% Wilson confidence intervals.

Algorithm	Rosenbrock Success Rate	MNIST Fluctuation Coefficient	Policy-Gradient Variance Reduction
Adam	100% (CI: 94.0% –100.0%)	0.418	Baseline
RMSProp	98.3% (CI: 91.1% –99.7%)	0.464	+10%
DSOF	**100% (CI: 94.0% –100.0%)**	**0.382**	**+38%**



(a) Convergence success rate.

(b) MNIST gradient variance.

Figure 5. Comparison of convergence success rates on the Rosenbrock function and gradient-variance fluctuation coefficients on MNIST.

As shown in Figure 6 and Table 4, the DSOF achieves the highest average episodic cumulative reward over the final five episodes. The DSOF obtains an average reward of −26.68, compared with −39.45 for Adam and −48.08 for RMSProp. The corresponding standard deviations are 11.95, 8.28, and 23.17, respectively. The results indicate that the DSOF achieves better final policy performance relative to Adam and RMSProp under the evaluated setting. The policy-gradient variance results in Table 3 are retained as a complementary measure of training stability.

The results demonstrate that the dynamic second-order momentum mechanism helps mitigate the effects of gradient noise and non-stationarity. In particular, the sliding-window estimator reduces

noise-induced variance fluctuations and enhances convergence stability in both synthetic and practical learning tasks.

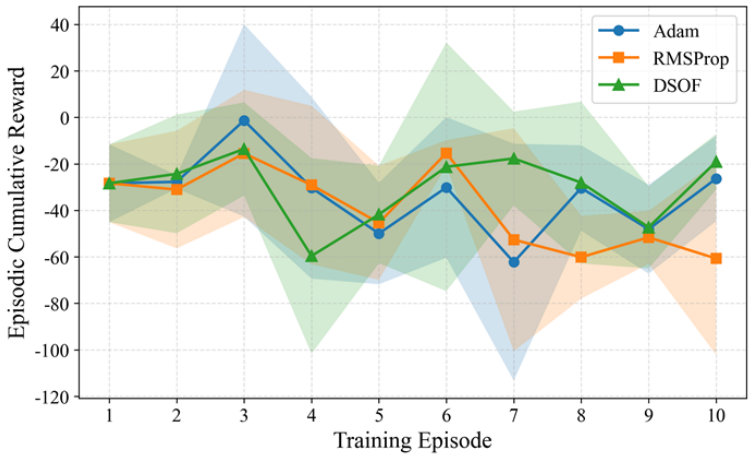


Figure 6. Episodic cumulative reward convergence on the HalfCheetah task.

Table 4. Average episodic cumulative reward over the final five episodes on the HalfCheetah task. Results are reported as mean ± standard deviation across independent runs.

Optimizer	**Final Episodic Cumulative Reward**
Adam	−39.45 ± 8.28
RMSProp	−48.08 ± 23.17
DSOF	−26.68 ± 11.95

3.6.2. Verification of the hybrid metaheuristic strategy

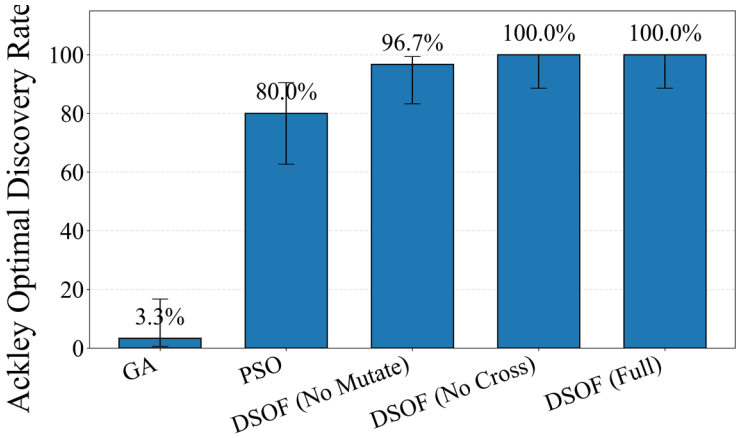
To evaluate the global-search capability of the DSOF against the GA and PSO, the hybrid metaheuristic strategy is examined through three complementary tasks. First, on the 50-dimensional Ackley function, the global optimum hit rate and the number of iterations to convergence are used to compare DSOF (Full), DSOF (No Mutate), DSOF (No Cross), the GA, and PSO in a highly multimodal landscape. Second, on the WikiText-2 dataset [27], a medium-scale transformer is trained and analyzed in terms of training-loss convergence and parameter diversity, where diversity is measured by cosine similarity among candidate solutions. This setting extends the evaluation beyond convolutional networks while remaining limited to the considered model and dataset configuration. Third, an ablation study is conducted by selectively disabling the crossover or mutation branch in Step 1.3.2 of Algorithm 1, so that the individual contributions of the two evolutionary operators to convergence efficiency and solution diversity can be isolated. The three tasks are therefore designed to separate global exploration, sequence-model optimisation behavior, and operator-level ablation, rather than to claim universal superiority over all black-box optimizers. The results are listed in Table 5.

In the Ackley test, DSOF (Full) achieves an optimum hit rate of 100%, compared with 3.3% for the GA and 80.0% for PSO, as shown in Table 5 and Figure 7(a). The corresponding 95% Wilson confidence intervals are 88.6%–100.0%, 0.6%–16.7%, and 62.7%–90.5%, respectively. These results indicate that the hybrid metaheuristic strategy achieves higher global-search reliability relative to the

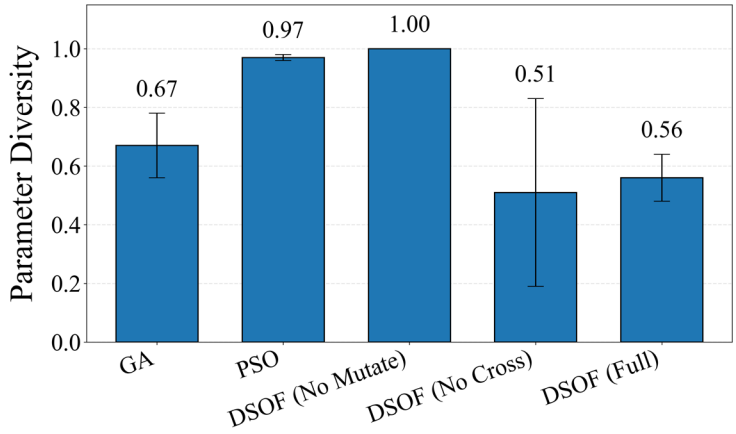
selected population-based baselines under the evaluated Ackley setting.

Table 5. Verification results of the hybrid metaheuristic strategy.

Method	Ackley Optimum Hit Rate (95% CI)	Iterations to Loss <2.0	Parameter Diversity
GA	3.3% (CI: 0.6–16.7%)	49 ± 0	0.67 ± 0.11
PSO	80.0% (CI: 62.7–90.5%)	52 ± 4	0.97 ± 0.01
DSOF (No Mutate)	96.7% (CI: 83.3–99.4%)	23 ± 6	1.00 ± 0.00
DSOF (No Cross)	100% (CI: 88.6–100.0%)	27 ± 12	0.51 ± 0.32
DSOF (Full)	**100% (CI: 88.6–100.0%)**	**19 ± 4**	**0.56 ± 0.08**



(a) Ackley optimum discovery rate.



(b) Parameter diversity comparison.

Figure 7. Comparison of the Ackley optimum hit rate and parameter diversity among the GA, PSO, DSOF, and its ablated variants.

The transformer-training results further show that DSOF (Full) reaches a loss below 2.0 after 19 ± 4 iterations, compared with 49 ± 0 iterations for the GA and 52 ± 4 iterations for PSO. Although DSOF (No Cross) and DSOF (No Mutate) also converge faster than the two baselines, the full configuration

requires the fewest iterations, indicating that the joint use of crossover and mutation contributes to improved convergence efficiency under the evaluated setting.

In the parameter-diversity analysis, DSOF (Full) obtains a diversity value of 0.56 ± 0.08, compared with 0.67 ± 0.11 for the GA and 0.97 ± 0.01 for PSO, as shown in Figure 7(b). Because the reported metric is based on cosine similarity, a lower value indicates lower parameter alignment and hence greater diversity. Therefore, DSOF (Full) exhibits lower parameter alignment and higher diversity than the GA and PSO during the evaluated transformer-training process.

The ablation results provide further insights into the contributions of the two evolutionary components. Removing mutation decreases the Ackley optimum hit rate from 100% to 96.7% and increases the number of iterations from 19 ± 4 to 23 ± 6. Removing crossover retains a 100% hit rate but increases the number of iterations to 27 ± 12. In addition, DSOF (No Cross) yields a parameter-diversity value of 0.51 ± 0.32, whereas DSOF (No Mutate) yields 1.00 ± 0.00. These results suggest that crossover primarily improves convergence efficiency and maintains effective exploration, while mutation contributes to search robustness and reduces the tendency toward excessive parameter alignment.

The experiments demonstrate the contribution of the HMSDE to improving global search and mitigating premature convergence in multimodal objective landscapes. The full DSOF configuration achieves the highest Ackley optimum hit rate, requires the fewest iterations to reach the specified transformer-training loss, and maintains lower parameter alignment than the GA and PSO under the evaluated transformer-training setting.

Relative to the selected GA and PSO baselines, the results demonstrate the value of periodically integrating evolutionary exploration into gradient-based optimization. The DSOF retains direct gradient updates and activates evolutionary operations only at specified intervals, making it applicable to the evaluated high-dimensional learning tasks such as transformer training.

3.6.3. Verification of the autonomous parameter space adjustment mechanism

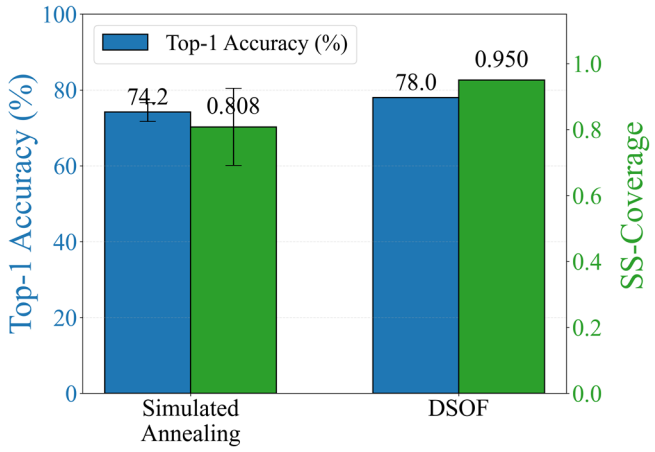
The autonomous parameter-space adjustment mechanism is evaluated against simulated annealing on three tasks covering image classification, path planning, and architecture search. On ResNet-50 image classification [28] and the ImageNet-1K dataset, the model is trained and the top-1 accuracy together with the dynamic boundary-adjustment trajectory is monitored throughout training. In the three-dimensional UAV path-planning environment with randomly generated dynamic obstacles, the average path length and the obstacle-avoidance rate are taken as the primary metrics under a fixed planning horizon and collision constraints. For the hyperparameter-optimization task [29], search-space coverage (SS-Coverage) is measured in a convolutional neural network (CNN) architecture search using the same candidate space for both methods. These three evaluations are intended to expose the trade-offs of the APSA. While boundary expansion and contraction can improve classification accuracy and coverage, the path-planning results also reveal that shorter paths do not necessarily correspond to higher obstacle-avoidance rates under the current single-objective boundary rule.

As shown in Table 6 and Figure 8(a), the DSOF achieves a top-1 accuracy of 78.0% in the ResNet-50 training task, compared with $74.2\% \pm 2.47\%$ for simulated annealing. This corresponds to a relative improvement of approximately 5.12% over simulated annealing. The result indicates that autonomous adjustment of the parameter search range can contribute to improved final classification

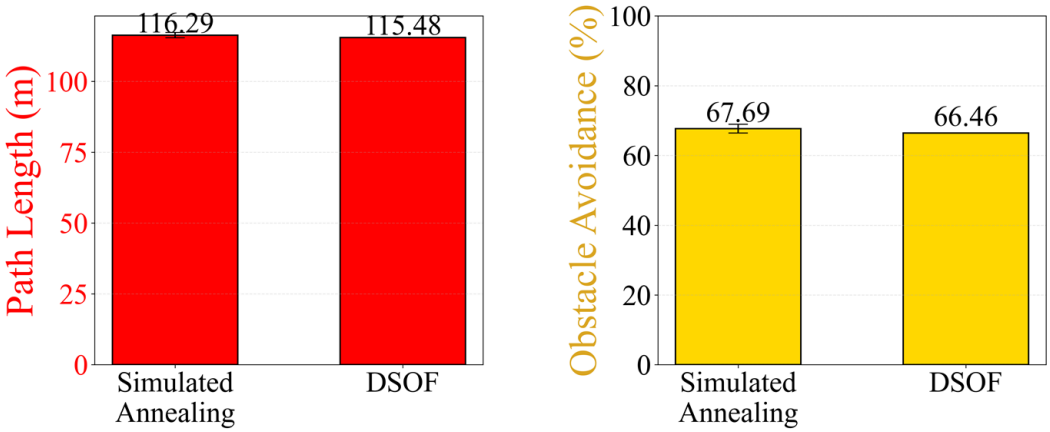
performance under the evaluated ImageNet-1K setting.

Table 6. Verification results of the autonomous parameter space adjustment mechanism.

Task	Metric	Simulated Annealing	DSOF	Improvement
ResNet Training	Top-1 Accuracy	74.2% ± 2.47%	78% ± 0.00	+5.12%
Path Planning	Avg. Path Length (m)	116.29 ± 0.93	115.48 ± 0.00	+0.69%
	Obstacle Avoidance	67.69% ± 1.25%	66.46% ± 0.00	−1.82%
Hyperparameter Opt.	SS-Coverage	0.808 ± 0.117	0.95 ± 0.00	+17.57%



(a) ResNet accuracy and SS-Coverage.



(b) UAV path-planning metrics.

Figure 8. Comparison of ResNet-50 top-1 accuracy, SS-Coverage, average path length, and the obstacle-avoidance rate between simulated annealing and the DSOF.

In the UAV path-planning task, the DSOF produces an average path length of 115.48 m, compared with 116.29 ± 0.93 m for simulated annealing, as shown in Table 6 and Figure 8(b). The resulting reduction of approximately 0.69% indicates that the DSOF generates slightly shorter paths under the evaluated environment, although the improvement is limited.

For obstacle avoidance, the DSOF obtains a rate of 66.46%, whereas simulated annealing

achieves 67.69% ± 1.25%. Thus, the DSOF is approximately 1.82% lower on this metric. This result indicates that the shorter paths generated by the DSOF do not necessarily lead to improved obstacle-avoidance performance under the current setting. Therefore, the path-planning results reflect a trade-off between the path length and obstacle-avoidance rate rather than consistent superiority across both metrics.

In the hyperparameter-optimization task, the DSOF achieves an SS-Coverage value of 0.95, compared with 0.808 ± 0.117 for simulated annealing. This represents an improvement of approximately 17.57% over simulated annealing, indicating that the adaptive parameter-space adjustment mechanism explores a larger proportion of the predefined search space.

The results confirm that the APSA supports dynamic adjustment of the parameter search range across image classification, UAV path planning, and hyperparameter optimization. The DSOF improves ResNet-50 top-1 accuracy, slightly reduces the average UAV path length, and increases search-space coverage. However, its obstacle-avoidance rate is slightly lower than that of simulated annealing under the evaluated setting. Overall, the APSA improves exploration and final performance on several metrics, while the path-planning results also reveal a trade-off that should be considered in future multi-objective optimization designs.

3.7. Comprehensive performance comparison

On NAS-Bench-201 [29], a tabular neural architecture search benchmark comprising 15,625 cell-based CNN architectures with precomputed training statistics, the DSOF demonstrates broad advantages over Adam and SMAC3, as detailed in Table 7.

Table 7. Performance comparison of Adam, SMAC3, and the DSOF on NAS-Bench-201. Resource costs are standardized in GPU-days.

Algorithm	Search Time (h)	Test Accuracy	Resource Cost (GPU-days)
Adam	118	93.3%	4.9
SMAC3	32	94.9%	1.3
DSOF	**25**	**93.63%**	**1.02**

The DSOF significantly improves search efficiency by reducing the search time from 118 hours for Adam to 25 hours, a reduction of 78.8%, and from 32 hours for SMAC3 to 25 hours, a reduction of 21.9%. The DSOF also lowers the resource cost to 1.02 GPU-days, representing reductions of 79.2% and 21.5% relative to Adam and SMAC3, respectively. In terms of accuracy, the DSOF achieves 93.63%, representing a 0.33-percentage-point improvement over Adam at 93.3% and a 1.27-percentage-point decrease relative to SMAC3 at 94.9%.

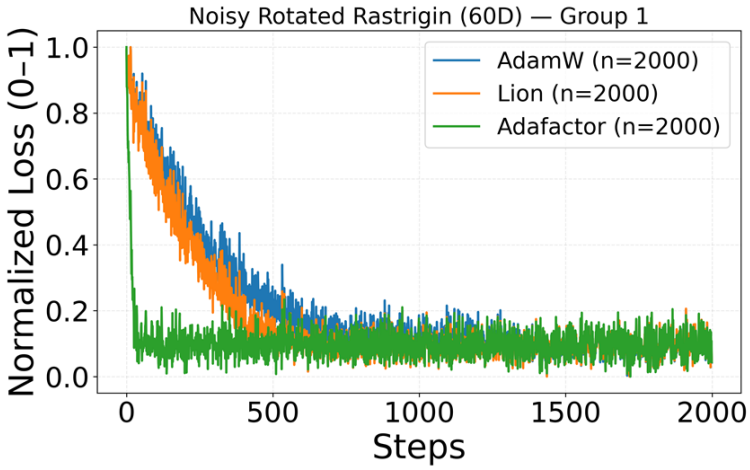
The comparison highlights the integrated advantages of the three DSOF mechanisms. By jointly improving stability through variance-aware momentum, global exploration through hybrid metaheuristics, and adaptability through boundary control, the DSOF achieves comparable accuracy with substantially lower time and resource costs than the selected baselines.

Overall, the results indicate that the DSOF addresses the main challenges identified in the problem formulation. Variance-aware momentum, hybrid metaheuristics, and dynamic boundary control contribute to robustness, diversity, and adaptability, respectively. The results support the design

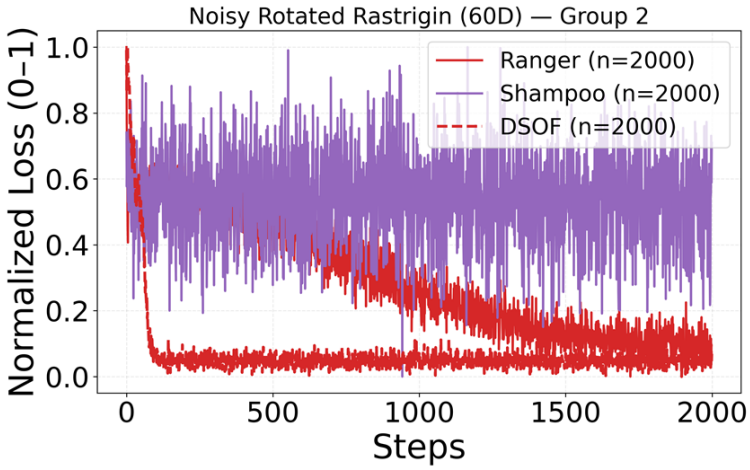
rationale and demonstrate the practical value of the DSOF in complex optimization applications.

3.8. Comparison with baseline optimizers

To evaluate the effectiveness of the DSOF, several representative optimizers are selected for comparison, including AdamW [30], Lion [31], Adafactor [32], Ranger [33], and Shampoo [34]. AdamW is a widely used adaptive optimizer with decoupled weight decay and has demonstrated robust performance across various deep learning tasks. Lion is a momentum-based first-order optimizer that provides rapid convergence with low memory consumption. Adafactor, designed for large-scale language models, reduces memory consumption through factorized second-moment estimates. Ranger combines Rectified Adam and Lookahead to improve generalization and stability. Shampoo is a second-order preconditioning method that accelerates convergence in high-dimensional optimization through block-wise matrix updates.



(a) Optimizer convergence performance on the 60-dimensional Rosenbrock function.



(b) Average runtime per optimization step.

Figure 9. Optimizer performance on the 60-dimensional Rosenbrock benchmark.

Figure 9 presents the normalized convergence curves of the DSOF and the baseline optimizers on the 60-dimensional Rosenbrock function. The DSOF achieves the fastest and most stable convergence among the evaluated methods. Adafactor exhibits rapid loss reduction during the early stage but slows during later iterations. AdamW and Lion show smooth convergence but require more iterations to reach low loss values. Shampoo maintains stable descent but converges more slowly, whereas Ranger exhibits the lowest convergence efficiency. In contrast, the DSOF rapidly reduces the loss during the initial iterations and maintains a stable trajectory with limited oscillation, ultimately achieving the best convergence performance.

The results demonstrate the advantages of the DSOF in convergence speed and stability, and indicate the potential of the DSOF for large-scale non-convex optimization.

As shown in Figure 10, the evaluated optimization algorithms exhibit substantial differences in computational efficiency. AdamW, Lion, Adafactor, and Ranger achieve an average step time of approximately 0.1 ms, demonstrating high computational efficiency and low per-iteration overhead. In contrast, Shampoo incurs a substantially higher runtime cost of nearly 1.8 ms. This increased cost mainly results from the matrix-preconditioning operations performed at each optimization step, which introduce additional computational and memory overhead. Although Shampoo can provide improved convergence performance in certain large-scale optimization scenarios, its higher computational cost may limit its efficiency in small-scale or time-sensitive applications.

The DSOF achieves a runtime comparable to AdamW and Adafactor, with an average step time of approximately 0.1 ms, indicating that the proposed dynamic variance-control mechanism introduces only limited additional computational overhead. These results demonstrate that the DSOF achieves a favorable trade-off between optimization performance and computational efficiency.

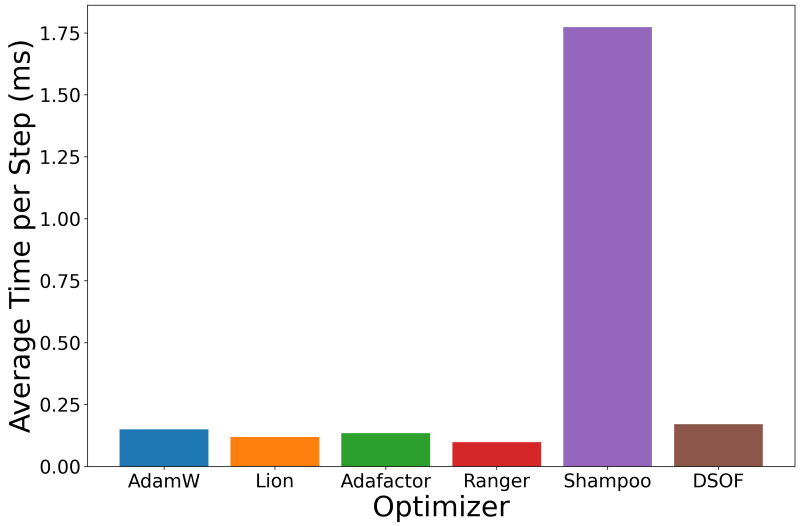


Figure 10. Average runtime per optimization step.

4. Discussion

The experimental results demonstrate that the proposed DSOF framework improves non-convex stochastic optimization not merely by increasing algorithmic complexity, but by coordinating three complementary mechanisms: variance-aware momentum estimation, diversity-enhanced global

exploration, and adaptive parameter-space regulation. The dynamic second-order momentum mechanism mainly contributes to stability by suppressing gradient-variance fluctuations and reducing noise-induced oscillations. Such improvements are particularly beneficial in noisy learning environments, including MNIST training and reinforcement learning policy optimization, where stochastic gradients may vary considerably across iterations. Compared with conventional EMA-based adaptive optimizers, the sliding-window variance estimator provides a more responsive representation of recent gradient behavior while avoiding excessive sensitivity to individual perturbations.

The hybrid metaheuristic strategy serves a different but complementary role. Gradient-based updates support efficient local descent but remain limited when the objective landscape contains multiple local optima or flat regions. By periodically introducing crossover and mutation operations, the DSOF maintains population diversity and improves the ability of the optimizer to explore alternative regions of the search space. The Ackley and Rosenbrock experiments indicate that the HMSDE is particularly useful in multimodal landscapes, where purely gradient-driven optimizers may converge prematurely. The ablation results further suggest that crossover contributes mainly to global exploration, whereas mutation helps preserve local diversity and prevents excessive concentration of candidate solutions.

The autonomous parameter space adjustment mechanism further enhances adaptability by modifying the search range according to the historical improvement rate. The APSA expands the feasible region when optimization progress stagnates and contracts the region during stable improvement. As shown in the UAV path-planning, ResNet training, and hyperparameter-optimization tasks, the APSA improves search efficiency and reduces unnecessary exploration. Therefore, the APSA provides an effective connection between global exploration and local refinement by adjusting the search range according to observed optimization progress.

Overall, the three DSOF components address complementary aspects of non-convex stochastic optimization. The DSMM improves robustness against stochastic gradient noise, the HMSDE enhances the ability to escape local optima, and the APSA adapts the search space to non-stationary or evolving landscapes. The integration of the three mechanisms enables the DSOF to balance convergence speed, optimization stability, solution diversity, and computational efficiency. Such coordination explains the competitive performance of the DSOF across heterogeneous tasks, including benchmark-function optimization, neural network training, reinforcement learning, neural architecture search, and UAV path planning.

Nevertheless, several limitations remain. First, the DSOF introduces additional hyperparameters, including the gradient-window size, metaheuristic interval, mutation strength, and boundary-adjustment thresholds. Although the sensitivity analysis indicates stable performance under a range of settings, broader evaluation across different problem classes remains necessary. Second, the population-based component may introduce additional memory overhead when the parameter dimension or population size becomes very large. The current deep-learning evaluation covers convolutional neural networks and a medium-scale transformer task but does not include large-scale language-model pretraining or modern vision transformer training. Therefore, the empirical findings cannot be directly generalized to all large-scale transformer-based optimization settings. Third, the current study provides primarily empirical validation and does not establish a formal convergence guarantee for the DSOF. A rigorous convergence analysis of stochastic gradient updates,

variance-dependent learning rates, evolutionary operations, and adaptive boundary adjustment requires additional assumptions and remains an important direction for future work.

5. Conclusions and future work

In this work, a DSOF is proposed to address the challenges of noisy, non-convex, and multimodal stochastic optimization. By integrating variance-aware gradient regulation, hybrid metaheuristic exploration, and autonomous parameter-space adjustment, the DSOF provides a unified optimization framework that improves update stability, global exploration capability, and search adaptability. Specifically, the dynamic second-order momentum mechanism reduces the influence of stochastic gradient fluctuations through sliding-window variance estimation, the hybrid metaheuristic strategy enhances population diversity through evolutionary exploration, and the autonomous parameter space adjustment mechanism dynamically regulates the search range according to optimization progress.

Extensive experiments on benchmark functions and practical tasks, including image classification, reinforcement learning, transformer training, neural architecture search, and UAV path planning, demonstrate the effectiveness of the DSOF under the evaluated settings. The results show that the DSOF achieves improved gradient stability, reliable global-optimum discovery, faster convergence, and competitive computational efficiency compared with representative optimization methods. Furthermore, ablation studies verify the complementary contributions of the three proposed components and validate the effectiveness of integrating adaptive gradient optimization with metaheuristic strategies.

Future work will focus on extending the DSOF to larger-scale and more complex optimization scenarios, such as distributed learning and foundation-model training, while further investigating the theoretical properties and convergence guarantees.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Guangxi Province under Grant 2022GXNSFAA035618.

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. C. Jin, P. Netrapalli, R. Ge, S. M. Kakade, M. I. Jordan, On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points, *J. ACM*, **68** (2021), 1–29. https://doi.org/10.1145/3418526

2. K. Xue, C. Qian, L. Xu, X. Fei, Evolutionary gradient descent for non-convex optimization, in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, (2021), 3221–3227. https://doi.org/10.24963/ijcai.2021/443

3. A. Cutkosky, H. Mehta, F. Orabona, Optimal stochastic non-smooth non-convex optimization through online-to-non-convex conversion, in *Proceedings of the 40th International Conference on Machine Learning*, (2023), 6643–6670.

4. K. Verma, A. Maiti, WSAGrad: A novel adaptive gradient based method, *Appl. Intell.*, **53** (2023), 14383–14399. https://doi.org/10.1007/s10489-022-04205-9

5. G. Ioannou, T. Tagaris, A. Stafylopatis, AdaLip: An adaptive learning rate method per layer for stochastic optimization, *Neural Process. Lett.*, **55** (2023), 6311–6338. https://doi.org/10.1007/s11063-022-11140-w

6. K. Jiang, D. Malik, Y. Li, How does adaptive optimization impact local neural network geometry? in *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, (2023), 8305–8384. https://doi.org/10.52202/075280-0366

7. F. Kunstner, A. Milligan, R. Yadav, M. Schmidt, A. Bietti, Heavy-tailed class imbalance and why Adam outperforms gradient descent on language models, in *38th Conference on Neural Information Processing Systems (NeurIPS 2024)*, (2024), 30106–30148. https://doi.org/10.52202/079017-0948

8. G. Amulya, A. Jyothi, S. Dasari, E. Sandhya, R. Kumar K V, Performance analysis of ML and DL models: Impact of linear and non-linear optimizers on model efficiency, *Adv. Nonlinear Var. Inequalities*, **28** (2025), 234–250. https://doi.org/10.52783/anvi.v28.2300

9. N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, *Evol. Comput.*, **9** (2001), 159–195. https://doi.org/10.1162/106365601750190398

10. D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, J. Schmidhuber, Natural evolution strategies, *J. Mach. Learn. Res.*, **15** (2014), 949–980.

11. I. Loshchilov, F. Hutter, SGDR: Stochastic gradient descent with warm restarts, in *International Conference on Learning Representations*, (2017), 1–16.

12. D. Sarkar, A. Biswas, Complex preference analysis: A score-based evaluation strategy for ranking and comparison of the evolutionary algorithms, *Soft Comput.*, **29**, (2025), 1967–1980. https://doi.org/10.1007/s00500-025-10525-y

13. B. Ghimire, A. Mahmood, K. Elleithy, Hybrid parallel ant colony optimization for application to quantum computing to solve large-scale combinatorial optimization problems, *Appl. Sci.*, **13** (2023), 11817. https://doi.org/10.3390/app132111817

14. Y. Hao, C. Zhao, Y. Zhang, Y. Cao, Z. Li, Constrained multi-objective optimization problems: Methodologies, algorithms and applications, *Knowl.-Based Syst.*, **299** (2024), 111998. https://doi.org/10.1016/j.knosys.2024.111998

15. F. Nikbakhtsarvestani, S. Rahnamayan, M. Ebrahimi, Opposition-based multi-objective ADAM optimizer (OMAdam) for training ANN, in *2024 IEEE Congress on Evolutionary Computation (CEC)*, (2024), 1–10. https://doi.org/10.1109/CEC60901.2024.10612083

16. L. Shen, C. Chen, F. Zou, Z. Jie, J. Sun, W. Liu, A unified analysis of AdaGrad with weighted aggregation and momentum acceleration, *IEEE Trans. Neural Networks Learn. Syst.*, **35** (2024), 14482–14490. https://doi.org/10.1109/TNNLS.2023.3279381

17. M. Reyad, A. M. Sarhan, M. Arafa, A modified Adam algorithm for deep neural network optimization, *Neural Comput. Appl.*, **35** (2023), 17095–17112. https://doi.org/10.1007/s00521-023-08568-z

18. H. Sun, L. Shen, Q. Zhong, L. Ding, S. Chen, J. Sun, et al., AdaSAM: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks, *Neural Networks*, **169** (2024), 506–519. https://doi.org/10.1016/j.neunet.2023.10.044

19. E. Levin, J. Kileel, N. Boumal, The effect of smooth parametrizations on nonconvex optimization landscapes, *Math. Program.*, **209** (2025), 63–111. https://doi.org/10.1007/s10107-024-02058-3

20. R. Islamov, N. Ajroldi, A. Orvieto, A. Lucchi, Loss landscape characterization of neural networks without over-parametrization, in *38th Conference on Neural Information Processing Systems (NeurIPS 2024)*, (2024), 46680–46727. https://doi.org/10.52202/079017-1481

21. W. Zhang, L. Niu, D. Zhang, G. Wang, F. U. D. Farrukh, C. Zhang, HW-Adam: FPGA-based accelerator for adaptive moment estimation, *Electronics*, **12** (2023), 263. https://doi.org/10.3390/electronics12020263

22. O. Hospodarskyy, V. Martsenyuk, N. Kukharska, A. Hospodarskyy, S. Sverstiuk, Understanding the Adam optimization algorithm in machine learning, in *2nd International Workshop on Computer Information Technologies in Industry 4.0*, (2024), 1–14.

23. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, et al., OpenAI Gym, preprint, arXiv:1606.01540.

24. P. Wawrzyński, A cat-like robot real-time learning to run, in *Adaptive and Natural Computing Algorithms*, (2009), 380–390. https://doi.org/10.1007/978-3-642-04921-7_39

25. L. Abualigah, A. Diabat, R. A. Zitar, Orthogonal learning Rosenbrock's direct rotation with the gazelle optimization algorithm for global optimization, *Mathematics*, **10** (2022), 4509. https://doi.org/10.3390/math10234509

26. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in *Proceedings of the IEEE*, **86** (1998), 2278–2324. https://doi.org/10.1109/5.726791

27. S. Merity, C. Xiong, J. Bradbury, R. Socher, Pointer sentinel mixture models, in *International Conference on Learning Representations*, (2017), 1–15.

28. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, (2016), 770–778. https://doi.org/10.1109/CVPR.2016.90

29. X. Dong, Y. Yang, NAS-Bench-201: Extending the scope of reproducible neural architecture search, in *International Conference on Learning Representations*, (2020), 1–16.

30. I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in *International Conference on Learning Representations*, (2019), 1–18.

31. B. Liu, L. Wu, L. Chen, K. Liang, J. Zhu, C. Liang, et al., Distributed lion for communication efficient distributed training, in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, (2024), 18388–18415.

32. N. Shazeer, M. Stern, Adafactor: Adaptive learning rates with sublinear memory cost, in *Proceedings of the 35th International Conference on Machine Learning*, (2018), 4596–4604.

33. L. Wright, N. Demeure, Ranger21: A synergistic deep learning optimizer, preprint, arXiv:2106.13731.

34. V. Gupta, T. Koren, Y. Singer, Shampoo: Preconditioned stochastic tensor optimization, in *Proceedings of the 35th International Conference on Machine Learning*, (2018), 1842–1850.

AIMS Press