Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

NEPHILA: A Bio-Inspired Optimization Algorithm Based on Adaptive Interception Dynamics and Learning-Guided Search

Gilberto Pérez Lechuga

gilberto.pl@cdmadero.tecnm.mx

Research Article

Keywords: Bio-inspired optimization, Adaptive interception dynamics, Learning-guided search, Reinforcement learning, Machine learning-assisted optimization, Stochastic search

Posted Date: November 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11368464/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

NEPHILA: A Bio-Inspired Optimization Algorithm Based on Adaptive Interception Dynamics and Learning-Guided Search

Gilberto Pérez-Lechuga[1*]

[1]División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero, Tecnológico Nacional de México, Av. 1º de Mayo y Sor Juana I. de la Cruz S/N, Col. Los Mangos, Ciudad Madero, 89440, Tamaulipas, México.

Corresponding author(s). E-mail(s): gilberto.pl@cdmadero.tecnm.mx;

Abstract

This paper introduces the NEPHILA Optimization Algorithm (NOA), a bio-inspired stochastic optimization framework motivated by the adaptive interception behavior of orb-weaving spiders. Rather than reproducing a literal biological model, NOA translates these dynamics into a learning-guided search architecture integrating Brownian perturbations, exponentially decaying Lévy flights, fitness-based adaptive pruning, elitist guidance, reinforcement-guided operator control, and a non-parametric Gaussian radial basis function model for estimating spatial solution quality.

The proposed framework was investigated through continuous and combinatorial optimization experiments. Six standard continuous benchmark functions were considered, with 30 independent runs per function and comparisons against PSO, ACOR, FA, BA, GWO, and Differential Evolution. The results revealed problem-dependent performance, with GWO obtaining the lowest mean error on five functions and PSO performing best on Schwefel.

To investigate applicability to discrete optimization, a permutation-based NEPHILA variant incorporating adaptive operator selection, edge-guided candidate selection, and bounded 2-opt improvement was evaluated on synthetic and TSPLIB Traveling Salesman Problem instances. On a synthetic 100-city instance, NEPHILA achieved the lowest mean tour length among the evaluated methods. However, comparisons with a GA-2opt baseline on four TSPLIB instances showed that NEPHILA obtained a lower mean tour length on one instance, while GA-2opt performed better on the remaining three.

Additional experiments demonstrated that increasing the evaluation budget from 15,030 to 60,000 reduced the mean tour length by approximately 29.69% on `kroA100`. Ablation results further revealed that simplified operator configurations could improve solution quality, although differences in local-search effort prevented attributing these gains exclusively to individual operators.
Overall, the findings establish NEPHILA as a flexible learning-guided stochastic optimization framework whose effectiveness depends on the objective landscape, computational budget, and coordination of search mechanisms. Rather than demonstrating universal superiority, the study identifies the conditions and algorithmic components that influence performance and provides an empirical basis for further refinement.

Keywords: Bio-inspired optimization, Adaptive interception dynamics, Learning-guided search, Reinforcement learning, Machine learning-assisted optimization, Stochastic search

MSC Classification: 90C26 , 90C59 , 68T05 , 68W20

1 Introduction

Orb-weaving spiders of the family *Araneidae* exhibit highly adaptive web-construction behavior that responds to environmental conditions such as prey traffic, wind direction, and energetic constraints [1, 2]. Species such as *Araneus* and *Trichonephila* continuously modify web orientation, asymmetry, and spiral density in order to improve prey interception while limiting silk expenditure [3]. Rather than actively pursuing prey, these spiders reorganize the geometry of the web itself, concentrating structural investment in regions where prey interactions are more likely to occur. This balance between adaptive resource allocation and probabilistic interception provides an interesting perspective for the development of optimization methods.

The literature reports numerous bio-inspired optimization methods based on the movement of agents toward candidate solutions through attraction, cooperation, or stochastic exploration mechanisms [4]. Approaches such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and related swarm-intelligence techniques have demonstrated strong performance in nonlinear and multimodal optimization problems [5–7]. However, biological systems in which search behavior can be interpreted through the adaptive organization of interception structures, rather than through direct pursuit or attraction mechanisms, remain comparatively less explored as a source of optimization strategies.

Motivated by these observations, this work introduces the **NEPHILA Optimization Algorithm (NOA)**, a bio-inspired optimization method based on the adaptive interception dynamics of orb-weaving spiders. The proposed approach combines probabilistic prey-density fields, Brownian local refinement, Lévy-flight global exploration, adaptive pruning, reinforcement-guided adaptation, and machine learning-based prey-field estimation within a unified stochastic optimization framework. In contrast to conventional swarm-based methods, NOA emphasizes adaptive spatial allocation over

direct agent attraction, allowing the search process to progressively concentrate exploration around regions associated with higher expected utility.

The proposed method is evaluated on six standard continuous optimization benchmarks—Sphere, Rosenbrock, Rastrigin, Ackley, Griewank, and Schwefel—through comparisons with PSO, ACOR, FA, BA, GWO, and Differential Evolution. Its applicability to combinatorial optimization is also examined through synthetic and TSPLIB Traveling Salesman Problem instances. These experiments assess solution quality, robustness, and computational performance across different optimization settings.

The main contribution of this work is a bio-inspired optimization framework that combines adaptive interception dynamics with stochastic exploration, learned spatial guidance, and reinforcement-guided search adaptation. The study further examines the behavior of these mechanisms through comparative experiments, sensitivity analysis, and ablation studies, identifying both their potential and their limitations across continuous and combinatorial optimization problems.

2 Literature Review

Nature-inspired optimization has become an established research area in computational intelligence, drawing on adaptive mechanisms observed in biological systems. Methods such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), the Firefly Algorithm, the Bat Algorithm, and the Grey Wolf Optimizer translate social interactions, attraction, echolocation, and cooperative behavior into computational search strategies [5, 6, 8–10]. The Dragonfly Algorithm, introduced by Mirjalili [11], provides another example of bio-inspired optimization, modeling static and dynamic swarming behaviors to regulate exploration and exploitation.

Most population-based approaches regulate exploration and exploitation through candidate-solution movement, stochastic perturbations, or information exchange. An alternative perspective is provided by orb-weaving spiders, whose prey-interception strategies involve adaptive modifications of web geometry and spatial organization rather than active pursuit. This behavior motivates optimization mechanisms based on the adaptive allocation of search effort across regions of the solution space.

Spider-related optimization methods include Social Spider Optimization (SSO), based on vibration-mediated cooperation [12]; Spider Monkey Optimization (SMO), inspired by the social behavior of spider monkeys [13]; and the Spider Wasp Optimizer (SWO), based on hunting, nesting, and mating behaviors [14]. Unlike these approaches, NOA draws on adaptive orb-web organization and probabilistic prey interception, integrating stochastic exploration, adaptive pruning, and learning-guided search to allocate computational effort across promising regions of the solution space.

3 Adaptive Exploration Dynamics of NOA

The NEPHILA Optimization Algorithm (NOA) draws inspiration from the adaptive web-construction behavior of orb-weaving spiders, which adjust web geometry in response to prey availability, environmental disturbances, and energetic constraints [1–3, 15]. Rather than simulating web structures, NOA translates these behaviors into

adaptive allocation of search effort: local adjustments motivate short-range refinement, broader reorganizations motivate global exploration, and reduced investment in unproductive regions motivates fitness-based pruning.

The resulting population-based framework combines Brownian perturbations, Lévy-flight exploration, adaptive pruning, reinforcement-guided operator control, learned spatial guidance, and elitist information. These mechanisms operate on candidate solutions through stochastic updates and objective-function evaluations, providing a computational interpretation of adaptive interception dynamics.

3.1 Mathematical Formulation

NOA is formulated as a population-based stochastic optimization method for continuous minimization problems. Let $f : D \subset \mathbb{R}^d \to \mathbb{R}$ denote the objective function to be minimized over a bounded search domain $D = [\ell, u]^d$, where $\ell < u$ denote the lower and upper bounds applied componentwise to all decision variables.

At iteration $t \in \{0, 1, \ldots, T\}$, the algorithm maintains a population

$$X^t = \left\{x_1^t, x_2^t, \ldots, x_N^t\right\}, \qquad x_i^t \in D, \tag{1}$$

where N is the population size and each vector $x_i^t = \left(x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,d}^t\right)$ represents a candidate solution.

The corresponding objective-function values are

$$f_i^t = f\left(x_i^t\right), \qquad i = 1, \ldots, N. \tag{2}$$

Since the problems considered in this study are formulated as minimization problems, the best-so-far objective value at iteration t is defined as

$$f_{\text{best}}^t = \min_{0 \le k \le t} \min_{1 \le i \le N} f\left(x_i^k\right), \tag{3}$$

and the associated best-so-far solution is

$$x_{\text{best}}^t \in \arg \min_{0 \le k \le t,\ 1 \le i \le N} f\left(x_i^k\right). \tag{4}$$

To characterize the relative quality of the current population, NOA converts the objective-function values into nonnegative normalized quality scores. Let

$$f_{\min}^t = \min_{1 \le i \le N} f_i^t, \tag{5}$$

and let $s_f^t > 0$ denote a scale estimate computed from the current objective-function values. The quality assigned to candidate x_i^t is

$$q_i^t = \exp\left[-\max\left(\frac{f_i^t - f_{\min}^t}{s_f^t}, 0\right)\right]. \tag{6}$$

Thus, the best current candidate receives quality $q_i^t = 1$, while candidates with larger objective-function values receive progressively smaller quality scores. These scores are used by the adaptive pruning mechanism and by the learned spatial quality model introduced below.

All candidate updates are projected onto the admissible search domain through

$$\Pi_D(z) = \min\{u, \max\{\ell, z\}\}, \tag{7}$$

where the minimum and maximum operations are applied componentwise.

To provide data-driven spatial guidance, NOA constructs a spatial quality field from an archive of previously evaluated candidate solutions. Let $\mathcal{A}_t = \{(x_i, f_i)\}_{i=1}^{M_t}$ denote the retained archive at iteration t, where M_t is the number of archived solutions. The quality scores q_i are computed using Equation (6), with the minimum objective value and normalization scale evaluated over the archived observations.

Using the resulting pairs (x_i, q_i), NOA estimates the spatial quality field through a Gaussian radial basis function model,

$$\widehat{P}(x) = \frac{\sum_{i=1}^{M_t} q_i \exp\left(-\dfrac{\|x - x_i\|^2}{2h^2}\right)}{\sum_{i=1}^{M_t} \exp\left(-\dfrac{\|x - x_i\|^2}{2h^2}\right) + \varepsilon}, \tag{8}$$

where $h > 0$ is the Gaussian kernel bandwidth and $\varepsilon > 0$ is a regularization constant that prevents numerical instability when the kernel weights become very small.

For the retained archive, let $F_t = \{f_i\}_{i=1}^{M_t}$ denote the set of archived objective-function values. The positive normalization scale is defined by

$$s_f = \max\left\{1.4826\ \mathrm{MAD}(F_t), \mathrm{Std}(F_t), 10^{-12}\right\},$$

where $\mathrm{MAD}(F_t)$ denotes the median absolute deviation from the median of the archived objective-function values, and $\mathrm{Std}(F_t)$ denotes their standard deviation. Consequently, the archived quality scores satisfy $q_i \in (0, 1]$, with larger values corresponding to better solutions.

The factor 1.4826 provides the conventional normal-consistency correction for the median absolute deviation (MAD), since $1/\Phi^{-1}(0.75) \approx 1.4826$, where Φ^{-1} denotes the standard normal quantile function. This scaling does not assume normally distributed objective values. Together with the standard deviation and a positive lower bound, it ensures a stable normalization scale.

The gradient $\nabla\widehat{P}(x, t)$ provides auxiliary directional guidance during the search and is rescaled to unit Euclidean norm whenever its norm exceeds one.

The learned field $\widehat{P}(x)$ is not a probability density. It represents estimated solution quality across the search domain, and its gradient guides the search toward regions predicted to contain better solutions.

NOA does not introduce a separate biological energy model. Instead, search adaptation is driven directly by the objective-function values and by a computational reward associated with candidate improvement and movement cost.

For a candidate transition from x_i^t to y_i^t, the normalized improvement is defined as $I_i^t = \max(f_i^t - f(y_i^t), 0)/\max(|f_i^t|, s_f^t, \varepsilon)$, where $s_f^t > 0$ is the current fitness scale and $\varepsilon > 0$ prevents numerical instability. The normalized movement cost is $C_i^t = \|y_i^t - x_i^t\|/(\sqrt{d}(u - \ell) + \varepsilon)$.

The reinforcement signal used by the adaptive policy is then

$$R_i^t = \text{clip}\left[\beta_I I_i^t - \beta_C C_i^t, -1, 1\right], \tag{9}$$

where $\beta_I > 0$ and $\beta_C > 0$ regulate the relative contributions of objective improvement and movement cost, respectively. Here, $\text{clip}(z, a, b) = \min\{\max\{z, a\}, b\}$ denotes the clipping operator, which bounds z to the interval $[a, b]$.

The optimization problem addressed by NOA is

$$x^* \in \arg\min_{x \in D} f(x), \tag{10}$$

where x^* denotes a global minimizer over the admissible domain D. NOA seeks improved approximations through adaptive stochastic search.

Candidate solutions are generated by combining Brownian local refinement, Lévy-flight global exploration, learned directional guidance, and elitist attraction, with adaptively regulated contributions. Each candidate is projected onto D and accepted only if it improves the current objective value. The stochastic operators and their integration into the candidate-update rule are described in the following subsections.

Brownian increments are sampled independently from $\Delta B_{i,t} \sim \mathcal{N}(0, I_d)$, whereas Lévy increments $\Delta L_{i,t}$ are generated using Mantegna's method with stability parameter $\beta = 1.5$. The Lévy intensity decreases exponentially according to

$$\sigma_{L,t} = \sigma_{L,0} \exp(-\kappa t), \tag{11}$$

where $\sigma_{L,0} > 0$ is the initial intensity and $\kappa > 0$ is the decay rate.

Adaptive exploration is controlled by a stochastic Softmax policy with four search modes: local refinement, balanced exploration, global exploration, and ML-guided exploration. Let $\phi = (\phi_1, \ldots, \phi_4)$ denote the policy logits. The probability of selecting mode j is

$$p_j = \frac{\exp(\phi_j/\tau)}{\sum_{k=1}^{4} \exp(\phi_k/\tau)}, \qquad j = 1, \ldots, 4,$$

The Softmax temperature $\tau > 0$ controls the stochasticity of action selection, with smaller values favoring higher-logit actions and larger values approaching uniform selection. In this implementation, $\tau = 1$, and actions are sampled as $a_i^t \sim \text{Categorical}(p_1, \ldots, p_4)$. Each action specifies multipliers for the Brownian, Lévy, and RBF-guidance components, whose values are reported in the experimental settings.

Following candidate evaluation, the policy logits are updated using the observed reward and a moving-average baseline, allowing the selection frequencies of the four exploration modes to adapt during optimization.

Adaptive pruning uses the previously defined quality scores q_i^t, with a linearly increasing pruning quantile level,

$$\eta_t = \eta_{\min} + (\eta_{\max} - \eta_{\min}) \frac{t}{T-1},$$

where $\eta_{\min}$ and $\eta_{\max}$ are the initial and final pruning quantile levels, respectively.

Let τ_t denote the empirical η_t-quantile of the current quality scores. Candidates satisfying $q_i^t \geq \tau_t$ remain active, while an elite fraction ρ_e is always preserved independently of the pruning threshold.

Inactive candidates are regenerated around randomly selected elite solutions according to

$$x_i^{t+1} = \Pi_D \left[x_e^t + \delta(u - \ell)\varepsilon_i^t \right], \qquad \varepsilon_i^t \sim \mathcal{N}(0, I_d),$$

where x_e^t is the selected elite solution and $\delta > 0$ controls the perturbation magnitude. The experimental values of ρ_e and δ are reported in the experimental settings.

The Softmax policy is updated after each evaluated action. Let $p = (p_1, \ldots, p_4)$ denote the action probabilities and $e_{a_i^t}$ the unit vector corresponding to the selected action. Using the moving-average reward baseline b_t, the advantage is $A_i^t = R_i^t - b_t$, and the logits are updated as

$$\phi \leftarrow \text{clip} \left[\phi + \eta_\phi A_i^t \left(e_{a_i^t} - p \right), \phi_{\min}, \phi_{\max} \right], \tag{12}$$

where $\eta_\phi > 0$ is the policy learning rate and clipping bounds the logits for numerical stability. The reward baseline follows

$$b_{t+1} = \lambda_b b_t + (1 - \lambda_b) R_i^t,$$

where $\lambda_b \in [0, 1)$ is the smoothing coefficient. This update adjusts the probability of each exploration mode according to its observed advantage.

3.2 Adaptive Stochastic Exploration Dynamics

NOA combines Brownian and Lévy perturbations to balance local refinement and broader exploration. These stochastic mechanisms operate directly on candidate solutions through discrete algorithmic iterations.

For each active candidate x_i^t, let $\varepsilon_{i,t}^B \sim \mathcal{N}(0, I_d)$. The Brownian perturbation is defined as

$$\Delta x_{i,t}^B = m_{B,a}\, \sigma_{B,t}\, c_B\, (u - \ell)\, \varepsilon_{i,t}^B,$$

where $\sigma_{B,t} > 0$ is the adaptive Brownian intensity, $m_{B,a} > 0$ is the multiplier associated with the selected search mode, and $c_B > 0$ scales the perturbation relative to the domain width.

The Brownian intensity is adjusted according to the mean reward from the preceding iteration:

$$\sigma_{B,t} = \text{clip} \left[\sigma_{B,0} \left(1 + \gamma_R \overline{R}_{t-1} \right), \sigma_{B,\min}, \sigma_{B,\max} \right], \tag{13}$$

where $\sigma_{B,0}$ is the reference intensity, $\overline{R}_{t-1}$ is the preceding mean reward, γ_R controls reward-based adaptation, and $\sigma_{B,\min}$ and $\sigma_{B,\max}$ specify the admissible intensity bounds.

The Brownian component provides short-range stochastic perturbations for local exploration, while Lévy flights introduce heavy-tailed increments that allow occasional longer exploratory steps. The use of such exploration mechanisms is consistent

with the analysis of random search strategies developed by Viswanathan et al. [16], who investigated the relationship between search efficiency and stochastic movement patterns.

For each active candidate, the Lévy vector is generated componentwise using Mantegna's method [17]:

$$L_{i,t} = \frac{U_{i,t}}{|V_{i,t}|^{1/\beta}},$$

where $U_{i,t} \sim \mathcal{N}(0, \sigma_u^2 I_d)$ and $V_{i,t} \sim \mathcal{N}(0, I_d)$ are independent Gaussian random vectors. The scale parameter is

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

where $\beta \in (0, 2)$ is the stability parameter.

The Lévy contribution to the candidate displacement is

$$\Delta x_{i,t}^{L} = m_{L,a}\,\sigma_{L,t}\,c_L\,(u - \ell)\,L_{i,t},$$

where $m_{L,a} > 0$ is the multiplier associated with the selected search mode, $c_L > 0$ is a domain-scaling coefficient, and $\sigma_{L,t}$ is the iteration-dependent Lévy intensity defined previously.

The increments generated by Mantegna's method exhibit asymptotic heavy-tailed behavior, $p(s) \sim |s|^{-1-\beta}, |s| \to \infty$, where $p(s)$ denotes the probability density of a signed increment s and $\beta \in (0, 2)$ is the stability parameter. This power-law decay allows occasional large displacements, complementing the shorter-range Brownian exploration mechanism.

For each active candidate x_i^t, the stochastic displacement combines the Brownian and Lévy contributions:

$$\Delta x_{i,t}^{\text{stoch}} = \Delta x_{i,t}^{B} + \Delta x_{i,t}^{L},$$

where the relative contributions are controlled by the selected Softmax search mode. The Lévy intensity decreases exponentially according to Equation (11). The resulting stochastic displacement is subsequently combined with RBF-based directional guidance and elitist attraction in the complete NOA update rule.

The Brownian exploration intensity is adjusted according to the mean reward from the preceding iteration, as defined in Equation (13). The clipping operation ensures that the intensity remains within the prescribed bounds.

4 Reinforcement Learning Framework

NOA employs a lightweight reinforcement-learning controller to adapt the selection of its search modes according to observed rewards [18]. The controller does not use an explicit state representation; instead, it maintains policy logits $\phi = (\phi_1, \ldots, \phi_K)$ for K available actions.

Action-selection probabilities follow the Softmax rule

Table 1 Action-dependent exploration multipliers used in NOA.

Search mode	$m_{B,a}$	$m_{L,a}$	$m_{M,a}$
Local refinement	1.35	0.55	0.80
Balanced exploration	1.00	1.00	1.00
Global exploration	0.65	1.45	0.75
ML-guided exploration	0.85	0.75	1.50

$$p_j = \frac{\exp(\phi_j/\tau)}{\sum_{k=1}^{K} \exp(\phi_k/\tau)}, \qquad j = 1, \ldots, K, \tag{14}$$

where $\tau > 0$ is the temperature parameter. At each decision step, an action is sampled from the categorical distribution $p = (p_1, \ldots, p_K)$.

4.1 Action Selection and Policy Update

NOA employs four search modes: local refinement, balanced exploration, global exploration, and ML-guided exploration. At each decision step, an action $a_i^t \in \mathcal{A} = \{a_1, \ldots, a_4\}$ is sampled according to the Softmax policy defined in Eq. (14):

$$a_i^t \sim \text{Categorical}(p_1, \ldots, p_4).$$

Each action determines the multipliers applied to the Brownian, Lévy, and RBF-guidance components, as reported in Table 1.

After evaluating the selected action, the policy logits are updated using the advantage $A_i^t = R_i^t - b$, where R_i^t is the observed reward and b is the moving reward baseline:

$$\phi \leftarrow \text{clip}\left[\phi + \eta_\phi A_i^t \left(e_{a_i^t} - p\right), \phi_{\min}, \phi_{\max}\right], \tag{15}$$

where $\eta_\phi > 0$ is the learning rate, $e_{a_i^t}$ is the unit vector associated with the selected action, and clipping bounds the logits for numerical stability. The baseline is updated after each reward observation as

$$b \leftarrow \lambda_b b + (1 - \lambda_b) R_i^t, \qquad \lambda_b \in [0, 1).$$

This update adapts action-selection probabilities according to their observed advantages. The corresponding parameter ranges are summarized in Table 1.

5 Algorithmic Implementation

This section describes the computational implementation of NOA, integrating stochastic exploration, learned RBF guidance, adaptive pruning, reinforcement-guided action selection, elitist attraction, and greedy acceptance within a population-based optimization framework.

5.1 General Optimization Procedure

Let $D \subset \mathbb{R}^d$ be the search domain and $X^t = \{x_1^t, \ldots, x_N^t\}$ the population at iteration t. NOA initializes and evaluates the population within D, retaining the best-so-far solution throughout the optimization process.

At each iteration, the algorithm updates the learned RBF quality field and identifies active candidates through adaptive pruning. Trial solutions combine Brownian perturbations with reward-adaptive intensity, Lévy perturbations with exponentially decaying intensity, RBF guidance, and elitist attraction. The Softmax policy determines the search-mode multipliers. Each trial solution is projected onto D and accepted only if it improves the current candidate. Experimental parameter values are reported in the Experimental Settings section.

At each iteration, the RBF quality field is updated using previously evaluated candidates, and adaptive pruning identifies active candidates. These candidates are processed according to the selected search mode, while inactive candidates are regenerated around randomly selected elite solutions. The resulting rewards update the Softmax policy and inform Brownian intensity adaptation in the next iteration. This process continues for T iterations, preserving the best-so-far solution. The complete pseudocode is provided in Appendix A.

6 Convergence and Complexity Analysis

This section establishes the best-so-far monotonicity property of NOA and examines its computational complexity. Global convergence is not claimed without additional assumptions.

6.1 Monotonicity of the Best-So-Far Objective Value

For the minimization problem, let $f_i^t = f(x_i^t)$ denote the objective value of candidate i at iteration t. The best-so-far value is defined as

$$f_{\text{best}}^t = \min_{0 \le k \le t} \min_{1 \le i \le N} f_i^k. \tag{16}$$

Since NOA preserves the best solution found and updates its record only upon improvement,

$$f_{\text{best}}^{t+1} \le f_{\text{best}}^t, \qquad t = 0, \ldots, T-1. \tag{17}$$

Hence, the best-so-far sequence is monotone non-increasing. The preservation of the best-so-far solution establishes a fundamental monotonicity property of NOA, formally

Proposition 1. For the minimization problems considered in this study, suppose that NOA retains the best objective-function value observed during the search. Then, the sequence $\{f_{\text{best}}^t\}_{t \ge 0}$ is monotone non-increasing; that is,

$$f_{\text{best}}^{t+1} \le f_{\text{best}}^t. \tag{18}$$

Proof Let $\mathcal{E}_t$ denote the set of candidate solutions evaluated during iteration t. By the best-so-far preservation rule,

$$f_{\text{best}}^{t+1} = \min\left\{ f_{\text{best}}^{t}, \min_{y \in \mathcal{E}_t} f(y) \right\}.$$

Since the minimum cannot exceed either argument,

$$f_{\text{best}}^{t+1} \leq f_{\text{best}}^{t}.$$

Hence, the sequence $\{f_{\text{best}}^{t}\}_{t \geq 0}$ is monotone non-increasing. □

The monotonicity established in Proposition 1 also yields a convergence result when the objective function is bounded below.

Corollary 1. Suppose that f is bounded below on D, such that $f(x) \geq f_{\text{inf}} > -\infty$ for all $x \in D$. Then the best-so-far sequence $\{f_{\text{best}}^{t}\}$ converges to a finite limit $\bar{f} \geq f_{\text{inf}}$.

Proof By Proposition 1, $\{f_{\text{best}}^{t}\}$ is monotone non-increasing and bounded below by f_{inf}. Hence, by the monotone convergence theorem for real sequences, $f_{\text{best}}^{t} \to \bar{f}$ for some finite $\bar{f} \geq f_{\text{inf}}$. □

The preceding result guarantees convergence of the best-so-far objective values to a finite limit, but neither global optimality nor probabilistic convergence to a global optimum is established for the implemented NOA.

6.2 Computational Complexity

Let N be the population size, T the number of iterations, d the search-space dimension, and M_A the maximum number of samples retained by the RBF model.

Brownian and Lévy perturbations, elitist attraction, projection, and greedy acceptance require $\mathcal{O}(d)$ operations per candidate. The Softmax controller adds constant overhead because the number of actions is fixed. Evaluating the Gaussian RBF guidance gradient requires $\mathcal{O}(M_A d)$ operations per candidate.

Thus, assuming that RBF-model maintenance and adaptive pruning do not exceed this cost, the overall search-update complexity is

$$\mathcal{O}(TNM_A d), \tag{19}$$

excluding the problem-dependent cost of objective-function evaluations. Since the RBF archive is bounded, the complexity is linear in T, N, and d for fixed M_A.

Adaptive pruning reduces the number of candidates requiring full search updates. Let $N_t^{\text{active}} \leq N$ denote the number of active candidates at iteration t. Their update cost is $\mathcal{O}(N_t^{\text{active}} M_A d)$, while regenerating each inactive candidate requires $\mathcal{O}(d)$ operations.

Defining the average active population as

$$\overline{N}_{\text{active}} = \frac{1}{T} \sum_{t=1}^{T} N_t^{\text{active}},$$

the cumulative active-candidate update cost is

$$\mathcal{O}\left(T\,\overline{N}_{\text{active}}\,M_A d\right). \tag{20}$$

Thus, adaptive pruning can reduce computational effort without changing the worst-case bound $\mathcal{O}(TNM_A d)$. Objective-function evaluation costs are excluded.

The computational savings depend on the average active-population size; therefore, no fixed asymptotic improvement is guaranteed.

6.3 Optimization Accuracy and Convergence Metrics

For benchmark functions with known global optimum f^*, optimization accuracy is measured by the mean absolute error across R independent runs:

$$\overline{\text{Error}} = \frac{1}{R}\sum_{r=1}^{R}\left|f_{\text{best}}^{(r)} - f^*\right|. \tag{21}$$

Here, $f_{\text{best}}^{(r)}$ denotes the final best objective value in run r. For the benchmarks considered, $f^* = 0$. Lower errors indicate greater optimization accuracy.

Run-to-run variability is measured by the sample standard deviation:

$$s_f = \sqrt{\frac{1}{R-1}\sum_{r=1}^{R}\left(f_{\text{best}}^{(r)} - \overline{f}_{\text{best}}\right)^2}, \tag{22}$$

where $\overline{f}_{\text{best}} = R^{-1}\sum_{r=1}^{R} f_{\text{best}}^{(r)}$. Smaller values of s_f indicate greater consistency across independent runs.

The error-based metrics quantify the proximity of the solutions to the known global optimum, whereas s_f characterizes the run-to-run variability of the final solutions.

6.4 Parameter Settings

Table 2 summarizes the parameter values used in the numerical experiments. These settings govern stochastic exploration, RBF-based guidance, adaptive pruning, reinforcement learning, reward computation, and elitist search.

The parameter values were selected through preliminary computational trials to balance exploration, exploitation, and search stability. Brownian and Lévy settings regulate local and global exploration, while adaptive pruning preserves population diversity and the bounded RBF archive controls computational cost. Reinforcement-learning parameters enable gradual policy adaptation. All settings were kept fixed across the benchmark functions to ensure a consistent experimental configuration.

7 Experimental Evaluation

NOA was evaluated on six benchmark functions: Sphere, Rosenbrock, Rastrigin, Ackley, Griewank, and Schwefel, representing unimodal and multimodal optimization landscapes. Its performance was compared with Particle Swarm Optimization (PSO), Ant Colony Optimization for Continuous Domains (ACOR), Firefly Algorithm (FA), Bat Algorithm (BA), Grey Wolf Optimizer (GWO), and Differential Evolution (DE).

Table 2 Parameter settings used in the NOA experiments.

Parameter	Description	Value
N	Population size	30
T	Number of iterations	500
$\sigma_{B,0}$	Baseline Brownian exploration intensity	0.05
$\sigma_{B,\min}$	Minimum Brownian exploration intensity	0.01
$\sigma_{B,\max}$	Maximum Brownian exploration intensity	0.10
$\sigma_{L,0}$	Initial Lévy exploration intensity	0.30
β	Lévy stability index	1.5
κ	Lévy exploration decay coefficient	0.01
γ_R	Brownian reward-adaptation coefficient	0.12
λ_M	Learned RBF-guidance coefficient	0.20
$\eta_{\min}$	Initial pruning quantile	0.15
$\eta_{\max}$	Final pruning quantile	0.45
ρ_e	Elite population fraction	0.20
η_ϕ	Policy learning rate	0.03
τ	Softmax temperature	1.0
λ_b	Moving-baseline decay coefficient	0.90
β_I	Reward weight for normalized improvement	1.0
β_C	Reward weight for normalized step cost	0.05
h_f	RBF bandwidth fraction	0.15
M_A	Maximum RBF archive size	240
d	Search-space dimensionality	30

All algorithms were evaluated in $d = 30$ dimensions using a population size of $N = 30$, $T = 500$ iterations, and $R = 30$ independent runs per benchmark. Optimization accuracy was assessed through the mean error relative to the known global optimum f^*, while run-to-run variability was measured using the sample standard deviation s_f, as defined previously.

An ablation study examined the contributions of Brownian exploration, Lévy flights, adaptive pruning, reinforcement-guided adaptation, and RBF-based guidance by individually removing each component.

For run r, the best-so-far objective value at iteration t is defined as

$$f_{t,\text{best}}^{(r)} = \min_{\substack{0\le k\le t\\ 1\le i\le N}} f\left(x_i^{k,(r)}\right). \tag{23}$$

Table 3 Benchmark functions used in the experimental evaluation ($d = 30$).

Function	**Mathematical Expression**	**Search Range**	f^*
Sphere	$f(x) = \sum_{i=1}^{d} x_i^2$	$[-100, 100]^d$	0
Rosenbrock	$f(x) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-30, 30]^d$	0
Rastrigin	$f(x) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right]$	$[-5.12, 5.12]^d$	0
Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^d$	0
Griewank	$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600, 600]^d$	0
Schwefel	$f(x) = 418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500, 500]^d$	0

The mean best-so-far trajectory across the R independent runs is $\overline{f}_{t,\text{best}} = R^{-1}\sum_{r=1}^{R} f_{t,\text{best}}^{(r)}$. This trajectory is used only to describe the empirical evolution of the best objective value during the search and does not constitute a theoretical global-convergence guarantee.

Table 3 summarizes the six benchmark functions used in the experiments. The set combines unimodal and multimodal problems with different landscape characteristics, providing a diverse test of local refinement and global exploration capabilities.

7.1 Experimental Setup

Each benchmark function was evaluated over $R = 30$ independent runs with dimension $d = 30$, population size $N = 30$, and $T = 500$ iterations. NOA was compared with PSO, ACOR, FA, BA, GWO, and DE.

Performance was assessed using the mean error relative to the known global optimum and the sample standard deviation of the final best objective values. The best-so-far objective value was also recorded during each run to describe the empirical evolution of the search.

All numerical calculations were performed using double-precision floating-point arithmetic. Table 4 summarizes the experimental configuration.

Table 5 reports the mean optimization error obtained by each algorithm. GWO achieved the lowest error on Sphere, Rosenbrock, Rastrigin, Ackley, and Griewank, whereas PSO performed best on Schwefel. NOA did not achieve the lowest error on

Table 4 Experimental configuration used for benchmark evaluation.

Setting	Value
Independent runs (R)	30
Population size (N)	30
Iterations (T)	500
Search-space dimensionality (d)	30
Stopping condition	Fixed number of iterations
Numerical precision	Double precision
Evaluated algorithms	NOA, PSO, ACOR, FA, BA, GWO, DE

Table 5 Mean optimization error over $R = 30$ independent runs. Lower values are better; the best result is shown in bold.

Algorithm	Sphere	Rosenbrock	Rastrigin	Ackley	Griewank	Schwefel
PSO	1.000×10^{3}	1.209×10^{4}	118.9	4.310	6.093	**3773**
ACOR	2.230×10^{-2}	222.7	231.2	4.202×10^{-2}	9.092×10^{-2}	8271
FA	4.809×10^{-13}	34.86	27.06	11.38	5.908×10^{-3}	9836
BA	66.85	1417	232.9	19.33	1.596	5499
GWO	$\mathbf{4.601 \times 10^{-31}}$	**27.20**	$\mathbf{1.895 \times 10^{-15}}$	$\mathbf{1.856 \times 10^{-14}}$	$\mathbf{2.216 \times 10^{-3}}$	1.118×10^{4}
DE	8.382	9287	137.9	2.539	0.4646	4907
NOA	0.6892	196.7	104.6	1.080	0.7773	4770

any benchmark under the current configuration, motivating the subsequent ablation analysis to examine the contribution of its principal search mechanisms.

Figure 1 shows the mean best-so-far trajectories for Rastrigin and Ackley over $R = 30$ independent runs, providing an empirical comparison of the search behavior of the evaluated algorithms.

The convergence curves show that GWO reaches lower best-so-far values on both Rastrigin and Ackley, whereas NOA improves more gradually and terminates at higher error levels. This suggests that NOA's search effectiveness depends on the optimization landscape and the interaction among its adaptive components.

To incorporate data-driven spatial guidance into the stochastic search process, the web-center update is formulated as

$$\begin{aligned} y_i^t = \Pi_D \Big[& x_i^t + \Delta x_{i,t}^B + \Delta x_{i,t}^L \\ & + m_{M,a} \lambda_M c_M (u - \ell) \nabla \widehat{P}_t(x_i^t) + c_E r_{i,t} \left(x_{\text{best}}^t - x_i^t \right) \Big], \end{aligned} \tag{24}$$

where $\widehat{P}_t$ denotes the learned RBF quality field, and $\nabla \widehat{P}_t(x_i^t)$ provides data-driven directional guidance. The terms $\Delta x_{i,t}^B$ and $\Delta x_{i,t}^L$ denote the Brownian and Lévy exploration components, respectively, while $m_{M,a}$ modulates the contribution of the learned guidance according to the search mode selected by the reinforcement-learning controller. The final term introduces attraction toward the best solution identified during the search.

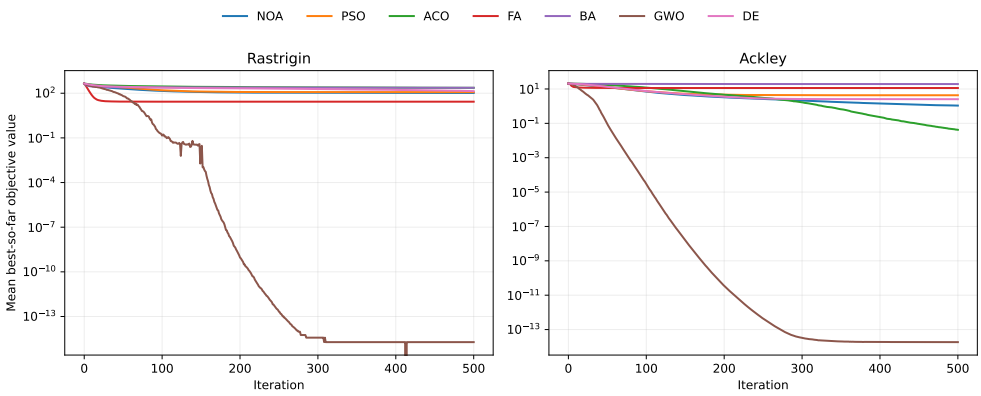


Fig. 1 Mean best-so-far objective trajectories obtained over $R = 30$ independent runs for the Rastrigin and Ackley benchmark functions. The figure compares the empirical convergence behavior of NOA with representative competing algorithms under the common experimental configuration of $N = 30$, $d = 30$, and $T = 500$. Lower objective values indicate better optimization performance. The trajectories illustrate differences in convergence rate and final solution quality among the considered methods without implying theoretical convergence to the global optimum.

The gradient $\nabla \widehat{P}_t(x)$ guides the search toward regions of higher estimated solution quality, while the Brownian and Lévy components preserve local and long-range stochastic exploration. Thus, the RBF model provides complementary data-driven guidance rather than replacing the stochastic search mechanisms. Its contribution is examined through the ablation experiments presented below.

From a biological perspective, this mechanism abstracts the adaptive use of accumulated environmental information in orb-weaving behavior rather than attempting to reproduce the underlying biological process literally.

Figure 2 summarizes the interaction among the principal adaptive mechanisms of NOA, including stochastic exploration, adaptive pruning, reinforcement-guided adaptation, and RBF-based search guidance.

These mechanisms operate jointly, and their contributions depend on the optimization landscape. Their individual effects are examined through the ablation analysis.

7.2 Ablation Study

An ablation study was conducted on Sphere, Rosenbrock, Rastrigin, Ackley, Griewank, and Schwefel to evaluate the contribution of the main NOA components. The variants remove Brownian exploration (NOA-noB), Lévy exploration (NOA-noL), adaptive pruning (NOA-noP), reinforcement learning (NOA-noRL), or RBF-based guidance (NOA-noML) from the complete algorithm (NOA-full).

Table 6 shows that the contribution of the different mechanisms is strongly landscape dependent. NOA-full achieves the lowest mean error on Ackley, whereas removing individual components produces better results on the other benchmark functions. In particular, removing Brownian exploration substantially improves the results

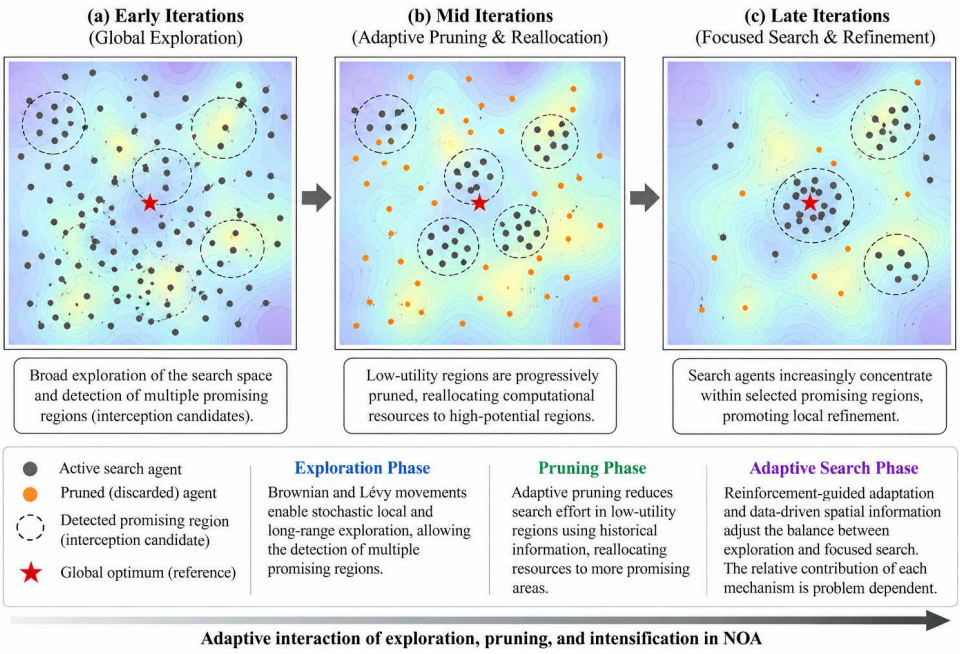


Fig. 2 Conceptual representation of the principal adaptive mechanisms of NOA. Brownian and Lévy perturbations support local and long-range exploration, adaptive pruning concentrates search effort, reinforcement learning adjusts the search behavior, and the RBF quality field provides data-driven guidance.

Table 6 Mean optimization error for the NOA ablation variants over $R = 30$ independent runs. Lower values are better.

Variant	Sphere	Rosenbrock	Rastrigin	Ackley	Griewank	Schwefel
NOA-full	0.7100	193.7	111.0	**0.7120**	0.7778	5070
NOA-noB	**0.006467**	391.0	100.1	1.971	**0.03963**	5226
NOA-noL	0.7401	514.0	100.3	2.772	0.7895	5659
NOA-noP	0.6997	**173.4**	**98.39**	4.644	0.7919	5812
NOA-noRL	1.029	241.1	108.4	0.9345	0.8591	**4900**
NOA-noML	0.7364	305.1	106.6	0.8003	0.7750	5068

Note: noB: without Brownian exploration; noL: without Lévy exploration; noP: without adaptive pruning; noRL: without reinforcement learning; noML: without RBF-based guidance. Bold values indicate the best result for each benchmark.

on Sphere and Griewank, while removing adaptive pruning yields the lowest error on Rastrigin. On Rosenbrock, NOA-noP also improves upon NOA-full, whereas on Schwefel the best result is obtained by NOA-noRL. These results indicate that the interaction among the NOA components is problem dependent and that no single mechanism provides a uniformly beneficial contribution across all benchmark landscapes.

The ablation results reveal substantial differences in the usefulness of the individual NOA mechanisms across optimization landscapes. Brownian exploration is beneficial

on Rosenbrock, Ackley, and Schwefel, but its removal produces pronounced improvements on Sphere and Griewank. Similarly, adaptive pruning improves performance on Ackley and Schwefel, whereas its removal gives the best results on Rosenbrock and Rastrigin. Reinforcement learning and RBF-based guidance also exhibit landscape-dependent effects, with the removal of reinforcement learning producing the lowest mean error on Schwefel. Overall, the results show that the contribution of the NOA components is not uniformly additive and suggest that their effectiveness depends on the current optimization landscape. This observation motivates the investigation of a state-aware adaptive mechanism capable of adjusting the contribution of the search components according to the observed search conditions.

8 Learning-Based Adaptive Exploration

NOA uses previously evaluated solutions to construct a local quality field that guides the search toward promising regions. This learned guidance complements the stochastic exploration mechanisms, with its effectiveness depending on the optimization landscape.

8.1 RBF-Based Quality-Field Estimation

Let $\mathcal{D}_t = \{(x_i, f_i)\}_{i=1}^{n_t}$ denote the bounded archive of previously evaluated candidate solutions, where $f_i = f(x_i)$ and the archive retains the best available solutions according to their objective values.

For a minimization problem, the archived objective values are transformed into quality scores according to

$$U_i = \exp\left[-\max\left(\frac{f_i - f_{\min}}{\max(s_f, 10^{-12})}, 0\right)\right], \tag{25}$$

where $f_{\min}$ is the minimum objective value in the retained archive and s_f is the scale computed from its objective values. Consequently, $U_i \in (0, 1]$, with larger values corresponding to better solutions.

The spatial quality field is estimated using Gaussian radial basis functions,

$$\widehat{P}(x,t) = \frac{\displaystyle\sum_{i=1}^{n_t} U_i \exp\left(-\frac{\|x - x_i\|^2}{2h^2}\right)}{\displaystyle\sum_{i=1}^{n_t} \exp\left(-\frac{\|x - x_i\|^2}{2h^2}\right) + 10^{-12}}, \tag{26}$$

where the bandwidth is defined as $h = \max\{(u - \ell)\alpha_h, 10^{-12}\}$, with α_h denoting the bandwidth fraction and $[\ell, u]$ the search interval.

The resulting field provides a smooth, data-driven estimate of solution quality based on the retained search history. Its gradient is used to construct an auxiliary search direction, with the gradient vector rescaled whenever its Euclidean norm exceeds one.

8.2 Hybrid Predictive Exploration

The RBF-based quality field guides candidate updates through Eq. (24), complementing the local Brownian and long-range Lévy perturbations. Its gradient directs the search toward regions of higher estimated solution quality, although improvements in the true objective function are not guaranteed because the estimate depends on the evaluated-solution archive. The guidance strength is controlled by the action-dependent multiplier $m_{M,a}$, the coefficient λ_M, the fixed scaling factor $c_M = 0.02$, and the domain width $(u - \ell)$, allowing its influence to vary with the selected search mode.

The update also includes elitist attraction toward the best-so-far solution, scaled by $c_E = 0.10$ and a random factor $r_{i,t} \sim U[0, 1)$. This introduces a stochastic bias toward the best-known position. Together, predictive guidance and stochastic perturbations balance the exploitation of promising regions with exploration of alternative solutions. Their effectiveness depends on the optimization landscape, as reflected in the ablation results.

Biologically, this mechanism is inspired by adaptive search behavior in which accumulated information influences subsequent decisions. It represents a computational abstraction rather than a literal model of orb-weaving spider behavior.

The RBF-based quality field complements Brownian and Lévy exploration through Eq. (24), directing candidate movements toward regions of higher estimated quality without guaranteeing objective improvement. Its influence depends on the evaluated-solution archive and is regulated by $m_{M,a}$, λ_M, $c_M = 0.02$, and the domain width $(u - \ell)$, according to the selected exploration mode. Elitist attraction toward the best-so-far solution is controlled by $c_E = 0.10$ and $r_{i,t} \sim \mathcal{U}[0, 1)$, preserving stochastic variability.

Together, these mechanisms combine predictive guidance, exploitation, and stochastic exploration, with landscape-dependent effectiveness reflected in the ablation results. Biologically, they represent a computational abstraction of adaptive search informed by previous evaluations, rather than a literal model of orb-weaving spider behavior.

8.3 Traveling Salesman Problem Application

The Traveling Salesman Problem (TSP) is a classical NP-hard combinatorial optimization problem that consists of determining a minimum-length closed tour visiting each city exactly once and returning to the initial city. Its mathematical structure provides a suitable framework for evaluating optimization algorithms designed to explore permutation-based solution spaces.

Let $G = (V, E)$ be a complete weighted graph, where $V = \{1, 2, \ldots, n\}$ denotes the set of cities and E represents the set of edges connecting distinct pairs of cities. Each edge $(i, j) \in E$ is associated with a nonnegative travel distance d_{ij}. In the symmetric Euclidean TSP considered in this study, $d_{ij} = d_{ji}$ and $d_{ii} = 0$.

For each ordered pair of distinct cities, define the binary decision variable

$$x_{ij} = \begin{cases} 1, & \text{if city } j \text{ is visited immediately after city } i, \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

The TSP can then be formulated as

$$\min_{x} \quad \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} x_{ij}, \tag{28}$$

subject to

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1, \qquad i \in V, \tag{29}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \qquad j \in V, \tag{30}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij} \leq |S| - 1, \qquad \forall S \subset V, \quad 2 \leq |S| \leq n - 1, \tag{31}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in V, \quad i \neq j. \tag{32}$$

The objective function (28) minimizes the total travel distance. Constraints (29) and (30) impose exactly one outgoing and one incoming edge at each city, respectively. Although these degree constraints ensure that every city is visited, they do not exclude solutions consisting of multiple disconnected cycles. The subtour-elimination constraints (31) prevent such configurations by prohibiting a closed cycle within any proper subset of cities. Together with the binary restrictions (32), these conditions characterize a single Hamiltonian tour.

NEPHILA-TSP 2.0 adapts the original optimization framework to the discrete space of feasible city permutations. Unlike continuous representations that require decoding into tours, each individual directly encodes a complete Hamiltonian cycle as an ordered sequence of cities.

Let

$$\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n) \in \mathcal{S}_n, \tag{33}$$

where $\mathcal{S}_n$ denotes the set of all permutations of n cities. The corresponding tour length is

$$L(\boldsymbol{\pi}) = \sum_{k=1}^{n-1} d_{\pi_k, \pi_{k+1}} + d_{\pi_n, \pi_1}. \tag{34}$$

The resulting optimization problem is

$$\boldsymbol{\pi}^* \in \arg\min_{\boldsymbol{\pi} \in \mathcal{S}_n} L(\boldsymbol{\pi}). \tag{35}$$

The 2-opt operator improves a tour by replacing two edges whenever the resulting configuration reduces its total length. For a symmetric TSP, replacing edges (a, b) and (c, d) with (a, c) and (b, d) produces the distance change

$$\Delta L = d_{ac} + d_{bd} - d_{ab} - d_{cd}. \tag{36}$$

A move is accepted when $\Delta L < 0$. Reversing the intermediate segment preserves permutation feasibility while enabling local exploitation of promising solutions.

Operator selection is governed by an adaptive softmax policy based on information accumulated during the search. Let $\mathcal{A}$ denote the available operators and $q_a^{(t)}$ their adaptive preferences at iteration t. The probability of selecting operator $a \in \mathcal{A}$ is

$$p_a^{(t)} = \frac{\exp\left(q_a^{(t)}/\tau\right)}{\sum_{b\in\mathcal{A}} \exp\left(q_b^{(t)}/\tau\right)}, \qquad a \in \mathcal{A}, \tag{37}$$

where $\tau > 0$ controls the concentration of the operator-selection probabilities.

The adaptive preferences are updated using a reward-based policy-gradient mechanism. Let a_t denote the selected operator, R_t its observed reward, and b_t a moving reward baseline. The advantage is defined as

$$A_t = R_t - b_t. \tag{38}$$

The preference associated with each operator $j \in \mathcal{A}$ is updated according to

$$q_j^{(t+1)} = \text{clip}\left[q_j^{(t)} + \frac{\alpha A_t}{\tau}\left(\mathbb{I}\{j = a_t\} - p_j^{(t)}\right), -8, 8\right], \tag{39}$$

where $\alpha = 0.03$ is the learning rate, $\tau = 1$ is the Softmax temperature, and $\mathbb{I}\{j = a_t\}$ is the indicator function of the selected operator.

The reward baseline is updated through

$$b_{t+1} = 0.9b_t + 0.1R_t, \qquad b_0 = 0. \tag{40}$$

The reward combines the relative improvement in tour length with a penalty for computational effort:

$$R_t = \text{clip}\left[\frac{\max(L_{\text{old}} - L_{\text{new}}, 0)}{\max(L_{\text{old}}, 10^{-12})} - 0.05\frac{C_t}{100}, -1, 1\right], \tag{41}$$

where C_t denotes the computational cost assigned to the selected operator.

NEPHILA-TSP 2.0 integrates candidate generation, solution evaluation, and adaptive operator selection. At each iteration, permutation-preserving operators generate alternative tours, evaluated using the total distance in Eq. (34). Their observed performance updates the adaptive selection preferences, allowing operator probabilities to evolve rather than follow a fixed schedule. This feedback balances exploration through larger tour modifications with exploitation through local improvement, without imposing a predetermined operator sequence.

The algorithm preserves the best feasible tour identified throughout the search. Whenever a candidate improves the best-known objective value, its permutation and tour length are recorded, ensuring that the best-so-far solution cannot deteriorate. The

Table 7 Comparative performance on the synthetic 100-city Euclidean TSP over 30 independent runs. All algorithms were subject to a maximum budget of 15,030 objective-function evaluations per run. Lower values indicate better performance.

Algorithm	Mean Distance (km)	Std. Dev. (km)	Mean Runtime (s)
PSO	6291.25	386.94	–
ACOR	7933.10	666.37	–
FA	8606.01	**76.85**	–
BA	7982.58	369.16	–
GWO	4195.24	351.36	–
DE	7396.36	792.34	–
NEPHILA-TSP 2.0	**3007.85**	118.37	11.34

search terminates upon exhausting the computational budget or satisfying another predefined stopping criterion, returning the shortest recorded tour.

NEPHILA-TSP 2.0 was evaluated on a synthetic 100-city Euclidean TSP instance against six metaheuristics: PSO, ACOR, FA, BA, GWO, and DE. Each method was executed 30 times with a population of 30 and a maximum of 15,030 objective-function evaluations per run. Performance was assessed through tour length, execution time, convergence behavior, and statistical significance. An ablation experiment also examined the effect of disabling the 2-opt operator.

Table 7 shows that NEPHILA-TSP 2.0 achieved the lowest mean tour distance (3007.85 km), outperforming GWO (4195.24 km) by approximately 28.3% and PSO (6291.25 km). Its standard deviation (118.37 km) indicates relatively consistent performance, although FA exhibited the lowest variability.

These findings support the effectiveness of the permutation-based approach on this instance. However, NEPHILA-TSP 2.0 includes additional 2-opt neighborhood evaluations, whereas the competing methods employ continuous random-key encodings. Thus, the observed performance differences cannot be attributed solely to the adaptive search mechanism.

Statistical analysis revealed significant differences among the seven algorithms (Friedman: $\chi_F^2 = 163.40$, $p = 1.13 \times 10^{-32}$). Pairwise Mann–Whitney tests with Holm correction confirmed significant differences between NEPHILA-TSP 2.0 and all competitors ($p_{\text{adj}} = 1.81 \times 10^{-10}$), supporting its superior observed solution quality on this instance. However, the Friedman test assumes comparable experimental blocks.

A 30-run ablation study showed that disabling 2-opt increased the mean tour distance from 3007.85 km to 6655.18 km (121.26% deterioration; Mann–Whitney $U = 0$, $p = 3.02 \times 10^{-11}$), while reducing mean runtime from 11.34 s to 7.25 s. This reveals a substantial trade-off between solution quality and computational cost. Because removing 2-opt also changes the adaptive operator-selection probabilities, its individual contribution cannot be isolated.

8.3.1 Ablation Study

An ablation study examined the contribution of 2-opt local search to NEPHILA-TSP performance. This operator improves feasible TSP tours by replacing two nonadjacent edges and reversing the intermediate segment whenever the resulting tour is

Table 8 Ablation analysis of NEPHILA-TSP on the 100-city Euclidean TSP ($R = 30$).

Performance measure	Complete	Without 2-opt
Mean distance (km)	**3007.85**	6655.18
Standard deviation (km)	**118.37**	179.26
Best distance (km)	**2745.13**	6204.87
Worst distance (km)	**3232.50**	6924.23
Mean runtime (s)	11.34	**7.25**
Mean full evaluations	14979.5	14970.0
Mean 2-opt checks	108780.7	0

shorter. Repeated improvements eliminate unfavorable connections while preserving the Hamiltonian cycle.

The complete algorithm was compared with a variant excluding 2-opt but retaining all other permutation-based operators. Both configurations were evaluated on the same 100-city Euclidean TSP instance over 30 independent runs, with a maximum budget of 15,030 complete objective-function evaluations per run.

Table 8 shows that disabling 2-opt increased the mean tour distance from 3007.85 km to 6655.18 km, an absolute difference of 3647.33 km (121.26%). Moreover, the worst result with 2-opt (3232.50 km) was substantially better than the best result without it (6204.87 km), indicating complete separation of the observed solution distributions.

The Mann–Whitney test confirmed this difference ($U = 0$, $p = 3.02 \times 10^{-11}$), with a rank-biserial effect size of 1.0. The complete algorithm also exhibited lower variability, with a standard deviation of 118.37 km versus 179.26 km without 2-opt, indicating improvements in both average solution quality and consistency.

These gains required additional computational effort. Mean runtime increased from 7.25 s to 11.34 s (a ratio of approximately 1.56), while the complete algorithm performed an average of 108780.7 additional 2-opt neighborhood checks per run, compared with none in the reduced variant. Despite similar numbers of complete objective-function evaluations, the computational workloads were therefore not equivalent.

Overall, the results provide strong empirical evidence favoring the complete configuration on the investigated TSP instance. However, disabling 2-opt also redistributes the adaptive selection probabilities among the remaining operators. Thus, the observed improvement reflects the combined effects of 2-opt inclusion and policy adaptation, rather than the isolated contribution of 2-opt under an unchanged selection policy.

8.4 Preliminary Comparison with a Discrete Genetic Algorithm

A preliminary comparison was conducted on the `eil51` TSPLIB instance, comprising 51 cities with a known optimal tour length of 426 distance units. NEPHILA-TSP 2.0 was compared against a permutation-based genetic algorithm (GA-2opt) employing order crossover, inversion mutation, tournament selection, elitist replacement, and bounded 2-opt local search.

Both algorithms used a population of 30, the same random seed (20261008), and a nominal budget of 15,030 complete objective-function evaluations. The number of examined 2-opt moves was also controlled to facilitate comparison of local-search effort.

The experiments followed the original TSPLIB `EUC_2D` convention, preserving integer edge weights and ensuring direct comparability with the published optimum.

Table 9 summarizes the results obtained from one independent execution of each algorithm.

Table 9 Preliminary comparison of NEPHILA-TSP 2.0 and GA-2opt on the TSPLIB `eil51` instance.

Performance indicator	**NEPHILA-TSP 2.0**	**GA-2opt**
Best tour length	**486**	499
Known optimal tour length	426	426
Optimality gap (%)	**14.0845**	17.1362
Complete objective evaluations	15,009	15,030
Examined 2-opt moves	132,056	132,056
Execution time (s)	12.874	**7.221**
Valid permutation	Yes	Yes

The optimality gap was computed as

$$\mathrm{Gap}(\%) = 100\,\frac{L_{\mathrm{alg}} - L^{*}}{L^{*}}, \tag{42}$$

where L_{alg} denotes the best tour length obtained by the corresponding algorithm and L^{*} represents the known optimal tour length.

NEPHILA-TSP 2.0 achieved a tour length of 486 (14.0845% optimality gap), compared with 499 (17.1362%) for GA-2opt. This represents an improvement of 13 distance units, or approximately 2.61% relative to GA-2opt. However, GA-2opt was faster, requiring 7.221 s versus 12.874 s, revealing a trade-off between solution quality and execution time.

Although both algorithms examined the same number of 2-opt moves, their different candidate-generation and population-update mechanisms imply that their computational workloads were not necessarily equivalent. Furthermore, NEPHILA-TSP 2.0 stopped after 208 generations because the remaining evaluation budget was insufficient, rather than reaching the configured maximum of 500 generations.

8.5 Statistical Comparison with GA-2opt on the TSPLIB `eil51` Instance

A statistical comparison of NEPHILA-TSP 2.0 and GA-2opt was conducted over 30 independent runs on the TSPLIB `eil51` instance (51 cities; known optimum: 426 distance units). GA-2opt employs order-preserving crossover, inversion mutation, tournament selection, elitist replacement, and bounded 2-opt local search.

Both algorithms used a population of 30, a nominal budget of 15,030 complete objective-function evaluations, and the same set of 30 random seeds for reproducibility.

The number of examined 2-opt moves was recorded to assess local-search effort. Tour lengths were computed using the original TSPLIB `EUC_2D` convention with integer-valued edge weights, ensuring comparability with the published optimum.

The optimality gap was calculated as

$$\mathrm{Gap}(\%) = 100 \left(\frac{L_{\mathrm{alg}} - L^*}{L^*} \right), \tag{43}$$

where L_{alg} denotes the best tour length obtained by a given algorithm in an individual execution, and L^* represents the known optimal tour length.

Table 10 presents the descriptive statistics obtained from the 30 independent executions of each algorithm.

Table 10 Statistical comparison of NEPHILA-TSP 2.0 and GA-2opt on the TSPLIB `eil51` instance over 30 independent runs.

Performance indicator	NEPHILA-TSP 2.0	GA-2opt
Number of runs	30	30
Mean tour length	**473.80**	484.30
Standard deviation	**8.54**	15.74
Best tour length	458	**457**
Worst tour length	**488**	513
Mean optimality gap (%)	**11.22**	13.69
Mean execution time (s)	9.70	**6.39**
Mean objective evaluations	14,985	15,030
Mean 2-opt checks	130,760	132,056

Overall, Table 10 shows that NEPHILA-TSP 2.0 achieved better mean solution quality, lower variability, and a smaller optimality gap than GA-2opt. Although GA-2opt obtained a slightly better best tour and shorter execution times, NEPHILA-TSP 2.0 exhibited more consistent performance across the 30 runs. These results suggest a trade-off between solution quality and computational efficiency.

8.6 Comparative Evaluation on TSPLIB Benchmarks

NEPHILA-TSP 2.0 was evaluated on four TSPLIB instances with different dimensions and geometric configurations: `eil51`, `berlin52`, `kroA100`, and `ch150`, comprising 51, 52, 100, and 150 cities, respectively. The comparison employed GA-2opt, a permutation-based genetic algorithm combining order-preserving crossover, inversion mutation, tournament selection, elitist replacement, and bounded 2-opt local search.

Each algorithm was independently executed 30 times per instance (240 runs in total), using a population of 30, a nominal budget of 15,030 complete objective-function evaluations, and identical sequences of random seeds for reproducibility. The number of examined 2-opt moves was also recorded. For `berlin52`, `kroA100`, and `ch150`, the GA-2opt local-search budget was matched to the 2-opt checks performed by NEPHILA-TSP 2.0 in each corresponding run; for `eil51`, a fixed limit of 132,056 checks was used.

All instances followed the original TSPLIB `EUC_2D` convention with integer-valued edge weights, ensuring direct comparability with their known optimal tour lengths.

The relative optimality gap was calculated as

$$\text{Gap}(\%) = 100\frac{L - L^*}{L^*}, \tag{44}$$

where L is the obtained tour length and L^* is the known optimum for each instance.

Table 11 reports the mean tour lengths and standard deviations over 30 independent runs per algorithm.

Table 11 Solution-quality comparison between NEPHILA-TSP 2.0 and GA-2opt on four TSPLIB benchmark instances. Results are reported as mean ± standard deviation.

Instance	Cities	Optimum	NEPHILA-TSP	GA-2opt
`eil51`	51	426	**473.80 ± 8.54**	484.30 ± 15.74
`berlin52`	52	7542	8785.10 ± 173.50	**8619.47 ± 309.93**
`kroA100`	100	21282	43922.23 ± 1763.29	**40750.23 ± 1502.99**
`ch150`	150	6528	19162.37 ± 605.46	**16532.27 ± 634.21**

The results show landscape-dependent performance. On `eil51`, NEPHILA-TSP 2.0 achieved a lower mean tour length than GA-2opt (473.80 versus 484.30), representing a 2.17% reduction, with lower variability.

For `berlin52`, GA-2opt achieved a better mean tour length (8619.47 versus 8785.10), although NEPHILA-TSP 2.0 showed lower variability. GA-2opt also outperformed NEPHILA-TSP 2.0 on the larger instances: 40750.23 versus 43922.23 on `kroA100`, and 16532.27 versus 19162.37 on `ch150`.

The results indicate that NEPHILA-TSP 2.0 did not maintain a consistent advantage in solution quality across the benchmark instances. Table 12 reports the mean optimality gaps and execution times.

Table 12 Mean optimality gaps and execution times over 30 independent runs. Gap values are expressed as percentages, whereas execution times are reported in seconds.

Instance	NEPHILA Gap	GA Gap	NEPHILA Time	GA Time
`eil51`	11.22	13.69	9.70	6.39
`berlin52`	16.48	14.29	9.52	6.20
`kroA100`	106.38	91.48	10.24	5.92
`ch150`	193.54	153.25	11.11	5.78

Under the prescribed computational budget, both algorithms showed substantial deterioration in solution quality on the larger instances. Mean optimality gaps exceeded 90% on `kroA100` and 150% on `ch150` for both methods. Although GA-2opt generally outperformed NEPHILA-TSP 2.0 on these instances, neither algorithm approached the known optima under the experimental settings.

GA-2opt also achieved shorter execution times across all four instances. This suggests that the adaptive operator-selection and population-update mechanisms of NEPHILA-TSP 2.0 may introduce computational overhead without corresponding improvements in solution quality under the evaluated configuration.

8.6.1 Statistical Significance Analysis

Statistical differences in tour-length distributions were assessed independently for each benchmark instance using a two-sided Mann–Whitney U test. Table 13 presents the resulting statistics and p-values.

Table 13 Mann–Whitney U test results for the comparison between NEPHILA-TSP 2.0 and GA-2opt.

Instance	U statistic	p-value	Lower mean distance
`eil51`	252.5	3.562×10^{-3}	NEPHILA-TSP
`berlin52`	651.0	3.034×10^{-3}	GA-2opt
`kroA100`	819.0	5.092×10^{-8}	GA-2opt
`ch150`	900.0	3.020×10^{-11}	GA-2opt

All four comparisons were statistically significant at $\alpha = 0.01$, even after applying the Holm correction for multiple testing. These results provide statistical evidence of differences between the tour-length distributions produced by NEPHILA-TSP 2.0 and GA-2opt across all four benchmark instances.

However, the direction of these differences depends on the benchmark instance. NEPHILA-TSP 2.0 achieved a lower mean tour length on `eil51`, whereas GA-2opt performed better on the other three instances. Thus, the results do not establish the general superiority of NEPHILA-TSP 2.0 over GA-2opt.

8.6.2 Discussion and Implications

The experimental findings highlight both the potential and the limitations of NEPHILA-TSP 2.0. Its favorable performance on `eil51` suggests that adaptive operator selection can be effective for certain problem configurations. However, the results on `berlin52`, `kroA100`, and `ch150` show that this advantage does not necessarily extend to other instances. One relevant factor is the fixed computational budget: although the number of cities increased from 51 to 150, the nominal number of complete objective-function evaluations remained approximately constant, limiting the available search effort relative to the growing permutation space.

This constraint may partially explain the substantial optimality gaps observed on larger instances, although the experiments do not establish insufficient evaluation budget as their principal cause. Other potential factors include neighborhood-operator effectiveness, the balance between exploration and exploitation, and the dynamics of adaptive operator selection. Distinguishing their individual contributions requires further investigation, particularly through systematic evaluation-budget sensitivity analysis and operator-level experiments.

Overall, these findings provide a transparent empirical assessment of NEPHILA-TSP 2.0 while identifying specific opportunities for improvement. Accordingly, subsequent experiments will examine convergence behavior, computational-budget sensitivity, and the contributions of individual search operators, with particular emphasis on the `kroA100` instance.

8.7 Sensitivity to the Computational Budget

The sensitivity of NEPHILA-TSP 2.0 to computational effort was evaluated on the TSPLIB `kroA100` instance using three maximum budgets: 15,030, 30,000, and 60,000 complete objective-function evaluations. Each configuration was tested over 30 independent runs with identical random seeds. Population size and algorithmic parameters remained unchanged, while the maximum number of generations was increased to 2,500 to accommodate larger budgets.

Table 14 Budget sensitivity analysis of NEPHILA-TSP 2.0 on the TSPLIB `kroA100` instance.

Indicator	**15,030**	**30,000**	**60,000**
Mean tour length	43922.23	35801.30	30879.03
Standard deviation	1763.29	1336.43	935.80
Best tour length	40585	32396	28547
Mean optimality gap (%)	106.38	68.22	45.09
Mean runtime (s)	10.64	18.26	31.76
Mean generations	210.47	392.67	737.10

As shown in Table 14, increasing the evaluation budget substantially improved solution quality. The mean tour length decreased from 43922.23 to 30879.03 as the budget increased from 15,030 to 60,000 evaluations, representing a reduction of approximately 29.69%.

The standard deviation decreased from 1763.29 to 935.80, indicating greater consistency across independent runs. A strong negative association between evaluation budget and final tour length was also observed (Spearman's $\rho = -0.94234$), although this descriptive correlation should be interpreted cautiously because identical random seeds were used across budget configurations. These improvements increased computational cost, with mean runtime rising from 10.64 to 31.76 seconds.

Although larger budgets substantially reduced the mean optimality gap, solutions remained well above the known optimum. These findings suggest that limited computational effort contributes to performance degradation, while further operator-level analysis is needed to determine whether the remaining gap also reflects limitations in exploration and exploitation.

8.8 Ablation Analysis

An ablation study on the TSPLIB `kroA100` instance examined the contributions of three components of NEPHILA-TSP 2.0: heavy-tailed inversion, segment relocation,

and adaptive operator selection. Three variants were evaluated by removing each component individually, with adaptive selection replaced by uniform random selection in the third variant. Each configuration was tested over 30 independent runs with a nominal budget of 15,030 objective-function evaluations.

Table 15 Ablation results on the TSPLIB `kroA100` instance.

Configuration	Mean	Std. dev.	Gap (%)
Full NEPHILA	43922.23	1763.29	106.38
Without heavy-tailed inversion	40952.17	1358.93	92.43
Without segment relocation	39790.43	1052.63	86.97
Without adaptive selection	40886.93	1457.01	92.12

As shown in Table 15, all three simplified variants achieved lower mean tour lengths than the full algorithm. Removing segment relocation produced the best result, reducing the mean tour length by approximately 9.41%. This suggests that the current combination of operators and adaptive selection may not provide the most effective exploration–exploitation balance for this instance.

However, the simplified variants performed approximately 23–26% more 2-opt neighborhood checks than the full algorithm. Therefore, the improvements cannot be attributed solely to component removal, since local-search effort also changed. These findings identify operator selection and computational-effort allocation as priorities for further refinement of NEPHILA-TSP 2.0.

8.9 Overall Discussion, Limitations, and Scalability

The experimental results highlight both the capabilities and limitations of NEPHILA-TSP 2.0, which integrates permutation-based operators, adaptive operator selection, edge-guided candidate selection, and bounded local improvement within a stochastic optimization framework. Its effectiveness depends on the problem instance, computational budget, and distribution of search effort. Across four TSPLIB benchmarks, NEPHILA-TSP 2.0 achieved better average solution quality than GA-2opt on `eil51`, whereas GA-2opt performed better on `berlin52`, `kroA100`, and `ch150`, while requiring less execution time in all four cases. Thus, the proposed method did not demonstrate a general performance advantage under the original computational budget.

The budget sensitivity analysis showed that increasing the evaluation budget from 15,030 to 60,000 on `kroA100` reduced the mean tour length by approximately 29.69% and the mean optimality gap from 106.38% to 45.09%. The accompanying reduction in standard deviation indicated greater consistency, although at the expense of longer execution times. Similarly, the ablation study revealed that removing selected components could improve performance under the original budget. In particular, excluding segment relocation reduced the mean tour length by approximately 9.41%. However, the modified configurations performed more 2-opt neighborhood checks, preventing

these improvements from being attributed exclusively to operator removal. These findings emphasize the need to refine adaptive operator selection and the allocation of computational effort.

Regarding scalability, NEPHILA-TSP 2.0 was successfully applied to instances containing up to 150 cities. Nevertheless, the substantial optimality gaps on larger instances do not support claims of superior scalability. Moreover, the evaluation covered only four benchmarks and a limited range of dimensions, preventing conclusions about asymptotic computational behavior or performance on substantially larger problems.

A further methodological limitation concerns computational-budget accounting. Although the algorithms used similar numbers of complete objective-function evaluations, their local-search neighborhood checks differed across configurations. Because complete tour evaluations and incremental 2-opt checks involve distinct computational operations, future comparisons should employ more comprehensive measures of computational effort.

Despite these limitations, the study establishes a reproducible experimental framework for investigating adaptive operator selection in permutation-based optimization. Future refinements should focus on calibrating operator probabilities, allocating local-search effort, and improving stopping criteria. Overall, NEPHILA-TSP 2.0 represents an experimentally evaluated framework with promising features and identifiable limitations, providing a basis for further development without establishing superiority over existing optimization methods.

9 Discussion and Potential Applications

The experimental results indicate that NEPHILA's performance depends on the optimization landscape and the interaction among its search mechanisms. Continuous benchmarks and ablation experiments showed that individual components contribute differently across problems, suggesting that NEPHILA's principal contribution lies in its adaptive architecture rather than universal superiority over established metaheuristics. This architecture combines Brownian perturbations, decaying Lévy flights, adaptive pruning, reinforcement-guided adaptation, Gaussian RBF-based prey-field estimation, and elitist guidance to integrate stochastic exploration with information from previously evaluated solutions. However, its effectiveness remains problem dependent.

The combinatorial extension, NEPHILA-TSP, operates directly on city permutations through adaptive operator selection and local improvement. On the 100-city Euclidean Traveling Salesman Problem, it achieved the lowest mean tour distance (3007.85 km), outperforming GWO (4195.24 km) and the remaining five competitors. Removing 2-opt increased the mean distance to 6655.18 km, a deterioration of 121.26%, while reducing execution time. These findings highlight the importance of combining adaptive exploration with local intensification. Nevertheless, differences in solution representation, the competitors' continuous random-key encodings, and additional 2-opt neighborhood evaluations prevent attributing the improvements exclusively to the adaptive architecture. Further validation on additional

Table 16 Architectural comparison of the evaluated optimization algorithms.

Mechanism	PSO	ACOR	FA	BA	GWO	DE	NEPHILA
Population-based search	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Stochastic exploration	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Adaptive pruning	No	No	No	No	No	No	Yes
Reinforcement-guided adaptation	No	No	No	No	No	No	Yes
RBF-based spatial learning	No	No	No	No	No	No	Yes

TSP instances and comparisons with specialized discrete optimization methods are therefore necessary.

Potential applications include logistics, vehicle routing, resource allocation, sensor deployment, and autonomous multi-agent coordination. The continuous formulation may also be explored for nonconvex optimization and hyperparameter tuning in machine learning. However, these applications require problem-specific adaptations and independent experimental validation.

Finally, Table 16 summarizes the main architectural differences between NEPHILA and competing optimization methods, characterizing their implemented mechanisms rather than establishing an intrinsic performance hierarchy. Overall, the findings support the potential of adaptive bio-inspired optimization while emphasizing the need to select and configure search mechanisms according to each problem's mathematical structure.

The main limitations involve computational cost, parameter sensitivity, and generalizability. RBF-based learning introduces additional overhead, while adaptive search components exhibit problem-dependent effectiveness. Moreover, this TSP comparison is limited to a single synthetic instance with differing solution representations among competitors. Future work will address broader benchmark validation, self-adaptive parameter control, and constrained, dynamic, and multi-objective optimization.

10 Conclusions

This study introduced NEPHILA, a bio-inspired stochastic optimization framework based on adaptive interception dynamics, learning-guided search, and the coordinated use of complementary exploration and exploitation mechanisms. Its continuous formulation integrates Brownian perturbations, decaying Lévy flights, adaptive pruning, reinforcement-guided operator selection, Gaussian RBF-based spatial learning, and elitist guidance. The main methodological contribution lies in combining these mechanisms within an adaptive architecture capable of responding to different search conditions.

The continuous benchmark experiments demonstrated that NEPHILA's performance depends on the objective-function landscape and the interaction among its components. Comparisons with established metaheuristics and the corresponding ablation studies revealed that no single mechanism consistently dominates across all problems. These findings emphasize the importance of adaptive coordination rather than suggesting universal superiority over existing methods.

The framework was further extended to discrete combinatorial optimization through NEPHILA-TSP, which employs permutation-based operators, adaptive selection, and bounded 2-opt local improvement. On a synthetic 100-city Euclidean TSP instance, the algorithm achieved the lowest mean tour distance of 3007.85 km, compared with 4195.24 km for GWO. Disabling 2-opt increased the mean distance to 6655.18 km, demonstrating the importance of local intensification under the evaluated configuration. Nevertheless, differences in solution representations and computational effort limit direct attribution of these improvements to the adaptive architecture alone.

Further experiments with NEPHILA-TSP 2.0 on four TSPLIB instances revealed heterogeneous performance relative to GA-2opt. The proposed method achieved better average solution quality on one instance, whereas GA-2opt performed better on the remaining three. Budget sensitivity analysis on `kroA100` showed that increasing the evaluation budget from 15,030 to 60,000 reduced the mean tour length by approximately 29.69%. Ablation experiments also indicated that simplified operator configurations could improve performance, although changes in local-search effort prevent attributing these gains exclusively to individual components.

Taken together, these results establish NEPHILA as a flexible research framework for investigating adaptive stochastic optimization in continuous and discrete search spaces. Its current limitations include problem-dependent performance, computational overhead, parameter sensitivity, and the need for broader experimental validation. Future research will address cost-aware operator selection, self-adaptive parameter control, improved computational-budget allocation, and more effective coordination between exploration and exploitation. Additional directions include constrained, dynamic, and multi-objective optimization, as well as applications in logistics, resource allocation, autonomous systems, and machine learning.

Overall, NEPHILA's contribution extends beyond the solution of a particular optimization problem. It provides an adaptive methodological foundation for combining stochastic search, learning-guided decisions, and problem-specific improvement operators. Although its present results do not establish universal superiority, they identify promising mechanisms and concrete opportunities for developing more effective and versatile optimization algorithms.

Appendix A

NEPHILA Optimization Algorithm (NOA) with Reinforcement-Guided and RBF-Based Adaptive Exploration

1: **Input:** Objective function f, search domain $D = [\ell, u]^d$, population size N, maximum iterations T
2: Initialize $X^0 = \{x_i^0\}_{i=1}^N$ uniformly in D
3: Evaluate $f_i^0 = f(x_i^0), \quad i = 1, \ldots, N$
4: Set $x_{\text{best}} = \arg\min_{x_i^0} f_i^0$ and $f_{\text{best}} = \min_i f_i^0$
5: Initialize bounded archive $\mathcal{A}_0 = \{(x_i^0, f_i^0)\}_{i=1}^N$
6: Initialize Gaussian RBF prey-field estimator $\widehat{P}(x)$
7: Initialize stochastic Softmax policy π_ϕ
8: Set previous mean reward $\overline{R}_{-1} = 0$

for $t = 0$ to $T - 1$ **do**

Fit the RBF prey-field estimator using the current bounded archive $\mathcal{A}_t$

Compute adaptive Brownian intensity

$$\sigma_B(t) = \text{clip}\left[\sigma_{B,0}\left(1 + \gamma_R \overline{R}_{t-1}\right), \sigma_{B,\min}, \sigma_{B,\max}\right]$$

Compute decaying Lévy intensity

$$\sigma_L(t) = \sigma_{L,0} e^{-\kappa t}$$

Convert the current objective values into normalized quality scores

$$q_i^t = \exp\left[-\max\left(\frac{f_i^t - f_{\min}^t}{s_f^t}, 0\right)\right]$$

Compute the pruning quantile

$$q_t = q_{\min} + (q_{\max} - q_{\min})\frac{t}{T-1}$$

Define the active population according to the corresponding quality threshold

Retain the elite fraction of the population as active regardless of the pruning threshold

Freeze the current best position as x_{best}^t

for $i = 1$ to N **do**

if x_i^t is pruned **then**

Select an elite anchor x_e^t

Generate

$$y_i^t = \Pi_D\left[x_e^t + 0.02\,(u - \ell)\,\varepsilon_i^t\right], \qquad \varepsilon_i^t \sim N(0, I_d)$$

else

Sample an exploration action

$$a_i^t \sim \pi_\phi(a)$$

Obtain the corresponding multipliers (m_B, m_L, m_M)

Generate Brownian and Lévy perturbations

$$B_i^t \sim N(0, I_d), \qquad L_i^t \sim \text{Levy}(\beta)$$

Evaluate the learned spatial gradient

$$g_i^t = \nabla \widehat{P}(x_i^t)$$

Compute the stochastic and learned steps

$$\Delta_i^B = m_B \sigma_B(t) 0.02\,(u-\ell) B_i^t$$

$$\Delta_i^L = m_L \sigma_L(t) 0.01\,(u-\ell) L_i^t$$

$$\Delta_i^M = m_M \lambda_M 0.02\,(u-\ell) g_i^t$$

Compute mild elitist attraction

$$\Delta_i^E = 0.10\, r_i^t \left(x_{\text{best}}^t - x_i^t\right), \qquad r_i^t \sim U(0,1)$$

Generate candidate

$$y_i^t = \Pi_D \left[x_i^t + \Delta_i^B + \Delta_i^L + \Delta_i^M + \Delta_i^E\right]$$

end if
Evaluate $f_y = f(y_i^t)$
if $f_y < f_i^t$ **then**
    Accept candidate: $x_i^{t+1} = y_i^t,\ f_i^{t+1} = f_y$
else
    Retain current solution: $x_i^{t+1} = x_i^t,\ f_i^{t+1} = f_i^t$
end if
if $f_y < f_{\text{best}}$ **then**
    Update $x_{\text{best}} = y_i^t$ and $f_{\text{best}} = f_y$
end if
Compute normalized improvement

$$I_i^t = \frac{\max(f_i^t - f_y, 0)}{\max(|f_i^t|, s_f^t, \varepsilon)}$$

Compute normalized step cost

$$C_i^t = \frac{\|y_i^t - x_i^t\|}{\sqrt{d}\,(u-\ell) + \varepsilon}$$

Compute reward

$$R_i^t = \text{clip}\left[\beta_I I_i^t - \beta_C C_i^t, -1, 1\right]$$

if x_i^t is active **then**
    Update the Softmax policy using R_i^t and the sampled action a_i^t
end if
Add (y_i^t, f_y) to the search archive
end for

Restrict the archive to the prescribed bounded memory size
Set

$$\overline{R}_t = \frac{1}{N}\sum_{i=1}^{N} R_i^t$$

Record f_{best} as the best-so-far value at iteration $t+1$
end for
return $x_{\text{best}}, f_{\text{best}}$

Availability of data and materials

The datasets generated and analyzed during the current study, together with the Python implementations [19] used to execute NOA, NEPHILA-TSP 2.0, and the comparative algorithms, and to reproduce the reported experiments, are available from the corresponding author upon reasonable request.

All computational experiments were conducted using Google Colaboratory (Google Colab) [20]. The experimental configurations, numerical outputs, and corresponding implementation details can also be provided to facilitate reproducibility and independent verification of the reported results.

Competing Interests

The author declares that there are no competing interests.

Funding

No funding was received for this study.

Authors' contributions

The author conceived the study, developed the NEPHILA Optimization Algorithm and its mathematical formulation, designed its adaptation to the traveling salesman problem through NEPHILA-TSP 2.0, performed the theoretical analysis and computational experiments, implemented the Python code, analyzed and interpreted the numerical results, and prepared, reviewed, and approved the final manuscript.

Acknowledgements

The author gratefully acknowledges the Instituto Tecnológico de Ciudad Madero and the Tecnológico Nacional de México for their institutional support and for providing an academic environment conducive to the development of this research.

References

[1] Foelix, R. F.: *Biology of Spiders*. 3rd ed., Oxford University Press, Oxford (2011)

[2] Eberhard, W. G.: *Spider Webs: Behavior, Function, and Evolution*. University of Chicago Press, Chicago (2020)

[3] Vollrath, F., Selden, P.: The role of behavior in the evolution of spiders, silks, and webs. *Annual Review of Ecology and Systematics* **32**, 547–571 (2001)

[4] Yang, X.-S.: *Nature-Inspired Optimization Algorithms*. Elsevier (2014)

[5] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948 (1995)

[6] Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)

[7] Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Oxford (1999)

[8] Yang, X.-S.: Firefly algorithms for multimodal optimization. In: *Stochastic Algorithms: Foundations and Applications*, Springer, pp. 169–178 (2009)

[9] Yang, X.-S.: A new metaheuristic bat-inspired algorithm. In: *Nature Inspired Cooperative Strategies for Optimization*, Springer, pp. 65–74 (2010)

[10] Mirjalili, S., Mirjalili, S. M., Lewis, A.: Grey wolf optimizer. *Advances in Engineering Software* **69**, 46–61 (2014)

[11] Mirjalili, S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications* **27**(4), 1053–1073 (2016)

[12] Cuevas, E., Cienfuegos, M., Zaldívar, D., Pérez-Cisneros, M.: A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications* **40**(16), 6374–6384 (2013)

[13] Bansal, J. C., Sharma, H., Jadon, S. S., Clerc, M.: Spider monkey optimization algorithm for numerical optimization. *Memetic Computing* **6**(1), 31–47 (2014)

[14] Abdel-Basset, M., Mohamed, R., Chakrabortty, R. K., Ryan, M. J., Mirjalili, S.: Spider wasp optimizer: A novel meta-heuristic optimization algorithm. *Artificial Intelligence Review* **56**, 11675–11738 (2023)

[15] Herberstein, M. E.: *Spider Behaviour: Flexibility and Versatility*. Cambridge University Press, Cambridge (2011)

[16] Viswanathan, G. M., Buldyrev, S. V., Havlin, S., da Luz, M. G. E., Raposo, E. P., and Stanley, H. E. (1999). Optimizing the success of random searches. Nature, 401, 911–914. https://doi.org/10.1038/44831

[17] Mantegna, R. N.: Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. *Physical Review E* **49**(5), 4677–4683 (1994)

[18] Sutton, R. S., Barto, A. G.: *Reinforcement Learning: An Introduction*. 2nd ed., MIT Press, Cambridge (2018)

[19] Van Rossum, G., Drake, F. L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009).

[20] Google: Google Colaboratory. Google Research (2026). https://colab.research.google.com/