International Journal of Multidisciplinary and Current Research ISSN: 2321-3124
Research Article Available at: https://ijmcr.com

A Reinforcement Learning-Guided Aquila Optimizer with Adaptive Strategy Selection for Enhanced Optimization in Complex Search Spaces

Ojo Olufemi Samuel[1*], Ganiyu Waheed Oyekunle [2], Ayeni Joshua Ayobami [3], Oyediran Mayowa Oyedepo[4]

[1, 3, 4]Department of Computer Sciences, Faculty of Computing, Ajayi Crowther University, P.M.B 1066, Oyo (Nigeria).
[2] Gracelink Computech Ventures, Awumaro Street, Oroki Road, Oyo (Nigeria).

Received 01 Apr 2026, Accepted 27 Apr 2026, Available online 28 Apr 2026, **Vol.14, No.2 (Mar/Apr 2026)**

Abstract

This research introduces a Reinforcement Learning-Guided Aquila Optimizer (RLG-AO) designed to solve complex global optimization problems through adaptive strategy selection. Traditional Aquila Optimizer (AO) methods rely on fixed or iteration-based switching strategies, which reduce flexibility in dynamic environments and can lead to premature convergence. To address this issue, the developed approach integrates a reinforcement learning mechanism that dynamically selects search strategies based on real-time feedback during optimization. The process is modeled as a sequential decision-making task, where the RL component chooses suitable operators using state information of fitness improvement, population diversity, and overall search progress. The model's performance is assessed using benchmark functions like Sphere, Rastrigin, and Ackley. Results show that RLG-AO improves convergence speed, enhances solution quality, and increases robustness compared to the standard AO. Overall, adaptive strategy selection proves effective in boosting performance across challenging optimization scenarios.

Keywords: *Reinforcement Learning, Aquila Optimizer, RLG-AO, Adaptive Strategy Selection, Metaheuristic Optimization, Benchmark Functions, Optimization Algorithms*

Introduction

Optimization is a core component of scientific computing, engineering design, and real-world decision-making (Rahmani et al., 2024). However, many real-life optimization problems involve high dimensionality, nonlinear relationships, and complex multimodal search spaces, which make traditional deterministic methods insufficient or inefficient (Ji et al., 2023).

To address these challenges, metaheuristic algorithms have gained prominence as effective alternatives due to their flexibility and ability to produce near-optimal solutions within reasonable computational time (SS and HS, 2022).

Among recent advancements in this field, the Aquila Optimizer (AO), developed by Abualigah et al. (2021), has shown strong potential for solving complex optimization problems. Inspired by the hunting strategies of Aquila eagles, the algorithm applies multiple search mechanisms to balance global exploration and local exploitation (Abualigah et al., 2024). Despite its effectiveness, the standard AO relies on predefined or iteration-based rules to switch between these strategies. This rigid approach limits its adaptability and can increase the likelihood of premature convergence, especially in complex, dynamic, and multimodal optimization scenarios.

Reinforcement Learning (RL), rooted in behavioral psychology and animal learning theories, offers a powerful framework for adaptive decision-making through ongoing interaction with the environment (Oh et al., 2020). Unlike conventional optimization methods, RL does not rely on prior knowledge of the problem space; instead, it learns optimal strategies through iterative trial-and-error guided by feedback signals. Its ability to adapt in real time and continuously improve makes it a compelling approach for increasing the intelligence and autonomy of optimization algorithms (AlMahamid and Grolinger, 2021).

Although a growing body of research has focused on enhancing the Aquila Optimizer (AO), most existing improvements remain heuristic and static. For example, Wang et al. (2021) combined AO with Harris Hawks Optimization to speed up convergence, while Gao et al. (2022) incorporated mutation-based strategies to increase population diversity. In a similar vein, Abualigah et al. (2023) employed opposition-based learning and Lévy flight mechanisms to enhance exploration. Despite achieving notable performance improvements, these

*Correspondant Author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijmcr/v.14.2.16

methods largely depend on fixed or rule-based modifications and lack the ability to adapt dynamically during the optimization process.

These limitations highlight a critical research gap the lack of an intelligent, self-adaptive mechanism that can dynamically select suitable search strategies based on real-time feedback from the optimization process. As a result, existing AO variants may deliver suboptimal performance when applied to diverse and non-stationary problem environments.

To overcome this limitation, this study develops a Reinforcement Learning-Guided Aquila Optimizer (RLG-AO). The framework embeds an RL agent within the AO to facilitate adaptive selection of search operators, enhancing convergence efficiency and the robustness of solutions. The performance of the developed approach is assessed using standard benchmark optimization functions.

Related Work

Recent research has improved the Aquila Optimizer through hybridization and heuristic-based modifications. For instance, Wang et al. (2021) integrated AO with Harris Hawks Optimization to enhance convergence, although the approach depends on fixed switching rules. Gao et al. (2022) applied mutation strategies to balance exploration and exploitation, but without incorporating adaptive intelligence. Similarly, Abualigah et al. (2024) introduced opposition-based learning and Lévy flight mechanisms, which improved convergence performance but still relied on static heuristic techniques.

Despite these improvements, existing methods still lack dynamic adaptability. This underscores the necessity of incorporating learning-based techniques, such as reinforcement learning, to enable real-time guidance in the optimization process.

Methodology

Research Approach and System Workflow

In this approach, the optimization process is modeled as a sequential decision-making problem in which a reinforcement learning (RL) agent interacts with the optimization environment to develop an effective strategy-selection policy. The state space is characterized by key indicators of the search process, including fitness improvement and population diversity, while the action space comprises the set of available Aquila Optimizer (AO) search operators. A reward mechanism is defined to encourage actions that enhance solution quality and to discourage stagnation or poor performance. The process starts with initializing the population and algorithm parameters, after which iterative cycles are performed. In each cycle, the RL agent observes the current state, chooses a suitable search operator, and applies it to update candidate solutions. The updated solutions are then evaluated, and a reward is computed based on the resulting change in fitness. This feedback is used to update the RL policy, enabling continuous learning and adaptation throughout the optimization process. The procedure is repeated until a termination criterion is satisfied, such as reaching the maximum number of iterations or achieving convergence. Ultimately, the best solution obtained is reported as the final output.

The developed RLG-AO integrates reinforcement learning into the Aquila Optimizer to enable adaptive selection of search strategies based on optimization feedback.

Benchmark Functions

The algorithm is evaluated using:
Sphere Function – unimodal (convergence speed)
Rastrigin Function – multimodal (global search ability)
Ackley Function – nonlinear (robustness)

Standard Aquila Optimizer (AO)

AO simulates four hunting strategies:

Expanded Exploration
Narrowed Exploration
Expanded Exploitation
Narrowed Exploitation

The Aquila Optimizer is a nature-inspired metaheuristic based on the hunting strategies of the Aquila (eagle). It alternates between exploration and exploitation to locate the global optimum.

Let

$$X_i^t = [\, x_{i1}^t, x_{i2,}^t \ldots\ldots x_{id}^t] \quad 1$$

Be the position of the i-th candidate solution (hyperparameters + threshold) at iteration t, in a d-dimensional search space.

The global best solution is:

$$X_{best}^t \quad 2$$

1) Population initialization

$$X_i^0 = LB + rand(0,1) \times (UB - LB) \quad 3$$

Where

LB and UB are lower and upper bounds

2) Mean Position of the Population

$$X_M^t = \frac{1}{N}\sum_{i=1}^{N} X_i^t \quad 4$$

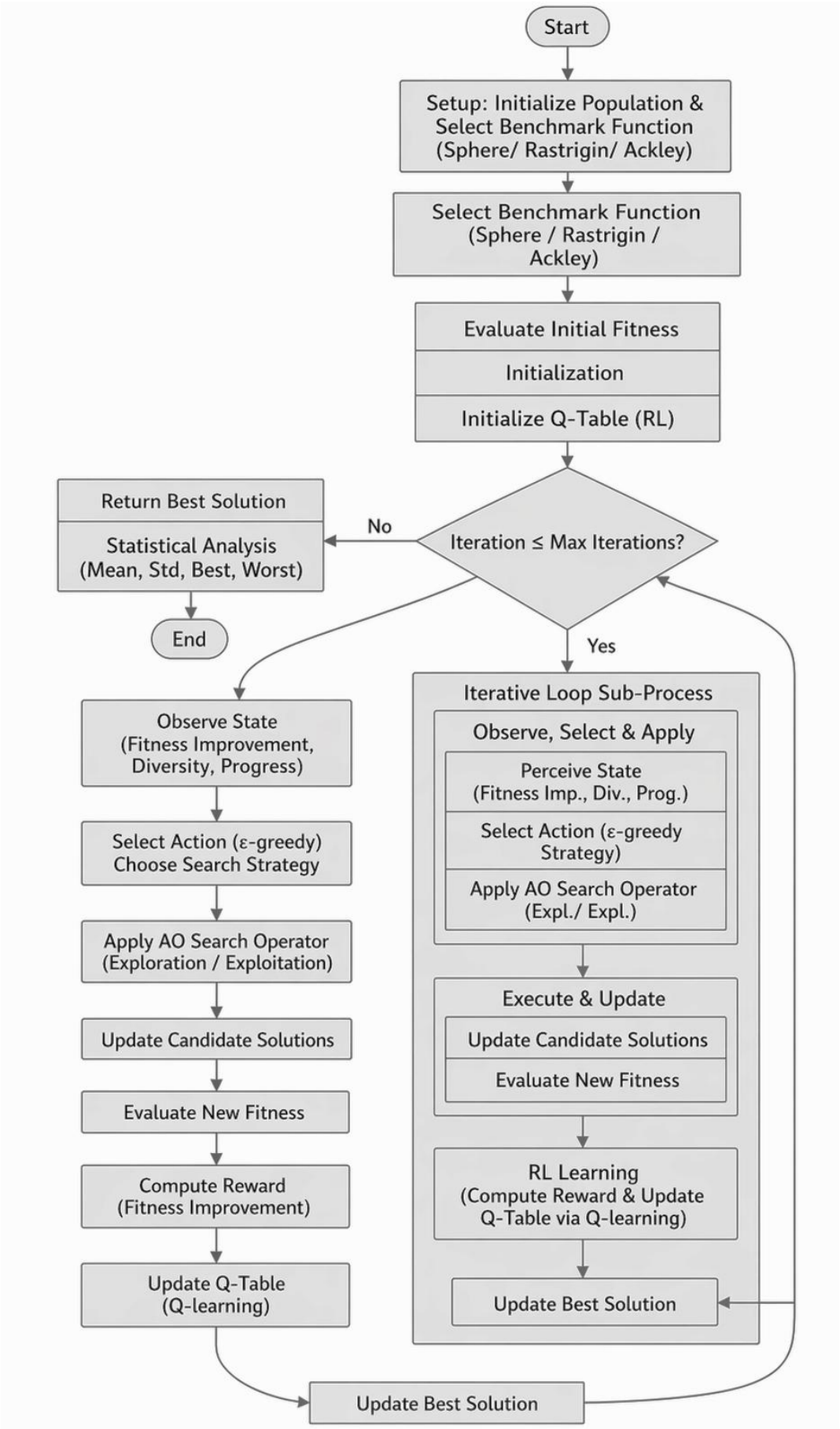

Figure 4: Flowchart of the Reinforcement Learning-Guided Aquila Optimizer (RLG-AO) with Adaptive Strategy Selection

4) Four hunting strategies of AO

AO uses four search operators to balance global and local search

A. Expanded exploration (High soaring)

Used in early iterations:

$$X_i^{t+1} = X_{best}^t(1 - \frac{t}{T}) + (X_M^t - X_{best}^t) .r \quad 5$$

This encourages global search of the hyperparameter space.

B. Narrowed exploration (Spiral flight)

$$X_i^{t-1} = X_{best}^t + S . (X_M^t - X_i^t) . r \quad 6$$

Where

$$S = 2 x (1 - \frac{t}{T}) \quad 7$$

Used to refine promising hyperparamter regions.

C. Expanded exploitation (Low altitude attack)

$$X_i^{t+1} = X_{best}^t + \propto . L . (X_i^t - X_{best}^t) \quad 8$$

Where:
L = Levy flight
$\propto$ = step size

Used for precise hyperparameter tuning.

D. Narrowed exploitation (Final caputure)

$$X_i^{t+1} = X_{best}^t + Q.(UB - LB) . r \quad 9$$

This performs local fine-tuning.

iv. Flight Levy

$$L = \frac{u}{|v|^{1/\beta}} \quad 10$$

Where:

$u \sim N(0, \sigma^2)$,
$v \sim N(0, 1)$,
$\beta = 1.5$

Levy flight enables AO to escape local optima.

The algorithm 1 begins by initializing a population X, where each individual represents a possible solution, along with control parameters such as α and δ, which regulate movement, randomness, and convergence. After initialization, the algorithm proceeds into an iterative loop that continues until a termination criterion is satisfied, such as reaching the maximum number of iterations T. In each iteration, the fitness of all candidate solutions is computed to assess their quality, and the best solution at that stage, $X_{best}(t)$, is identified. The algorithm then updates the population mean $X_M(t)$ along with several control parameters, including x, y, G_1, and the Lévy flight distribution, which enables long-distance movements to strengthen global exploration and reduce the risk of premature convergence. The Aquila Optimizer (AO) alternates between exploration and exploitation depending on the iteration index. Specifically, when $t \leq \frac{2}{3}T$, the algorithm prioritizes exploration by broadly scanning the search space. In this phase, it performs either Expanded Exploration (X1) or Narrowed Exploration (X2) depending on a random probability. Expanded exploration allows wide, global movement, while narrowed exploration performs more focused searching around promising regions.

When $t > \frac{2}{3}T$, AO shifts to exploitation, where it refines already promising solutions. This is achieved through either Expanded Exploitation (X3) or Narrowed Exploitation (X4). Expanded exploitation slightly perturbs the current best solutions to check nearby regions, while narrowed exploitation tightly fine-tunes the solution to converge toward the global optimum. After each update, the new solution is evaluated. If it improves upon the current solution or the global best X_{best}, it replaces them. This process continues until the stopping condition is met. Finally, the algorithm returns X_{best}, which represents the optimal solution found. In this research, best solution corresponds to the optimal hyperparameters.

The Pseudo-code of the AO is detailed in Algorithm

Algorithm 1 Aquila Optimizer (Abualigah *et al.,* 2021).

Initialization phase:
Initialize the population X of the AO.
Initialize the parameters of the AO (i.e., α, δ, etc).
WHILE (The end condition is not met) do
Calculate the fitness function values.
Xbest(t)= Determine the best obtained solution according to the fitness values.
for (i = 1,2…,N) do
Update the mean value of the current solution XM(t).
Update the x, y,G1,G2, Levy(D), etc.
if $t \leqslant (\frac{2}{3}) * T$ then
if rand$\leqslant$0.5 then
Step 1: Expanded exploration (X_1)
Update the current solution using Eq. (3).
if Fitness($X_1(t + 1)$) < Fitness(X(t)) then
X(t) =($X_1(t + 1)$)
if Fitness($X_1(t + 1)$) < Fitness(Xbest(t)) then
Xbest(t) =X1(t + 1)
end if

end if
else
{ ▹ Step 2: Narrowed exploration (X2)}
Update the current solution using Eq. (5).
if Fitness(X2(t + 1)) < Fitness(X(t)) then
X(t) =(X2(t + 1))
if Fitness(X2(t + 1)) < Fitness(Xbest(t)) then
Xbest(t) =X2(t + 1)
end if
end if
end if
else
if rand⩽0.5 then
{ ▹ Step 3: Expanded exploitation (X3)}
Update the current solution using Eq. (13).
if Fitness(X3(t + 1)) < Fitness(X(t)) then
X(t) =(X3(t + 1))
if Fitness(X3(t + 1)) < Fitness(Xbest(t)) then
Xbest(t) =X3(t + 1)
end if
end if
else
Step 4: Narrowed exploitation (X4)
Update the current solution using Eq. (14).
if Fitness(X4(t + 1)) < Fitness(X(t)) then
X(t) =(X4(t + 1))
ifFitness(X4(t + 1)) < Fitness(Xbest(t)) then
Xbest(t) =X4(t + 1)
end if
end if
end if
end if
end for
end while
return The best solution (Xbest).

Reinforcement Learning-Guided AO (RLG-AO)

An Enhanced Aquila Optimizer (RLG-AO) is formulated by integrating a reinforcement-learning–guided (RLG) mechanism. In the enhanced version, reinforcement learning dynamically controls the selection of AO operators and adapts algorithm parameters based on real-time optimization feedback. By learning from previous search experiences, the RLG-AO intelligently adjusts exploration and exploitation strategies, avoids ineffective operators, and improves convergence stability. Algorithm 2 describes the Reinforcement Learning–Guided Aquila Optimizer (RLG-AO), which enhances the standard Aquila Optimizer by introducing an adaptive learning mechanism. Instead of using fixed search strategies, a reinforcement learning agent observes the optimization state (such as fitness improvement, population diversity, and iteration progress) and dynamically selects the most suitable Aquila operator at each iteration. The population of candidate solutions is updated using the chosen operator, and their fitness values are re-evaluated. Based on quality improvement, a reward is calculated and used to update the RL policy. This learning-based selection mechanism enables the optimizer to more effectively balance exploration and exploitation, resulting in improved convergence speed and higher-quality solutions.

Algorithm 2: RLG-AO (Pseudocode)

Algorithm 2: Reinforcement Learning-Guided Aquila Optimizer (RLG-AO)

Input:
Objective function f(x)
Population size N
Maximum iterations T
Search space bounds (LB, UB)
RL parameters (α, γ, ε)
Output:
Optimal solution x_best
1: Initialize population X = {x1, x2, ..., xN} within bounds (LB, UB)
2: Initialize Q-table Q(s, a) arbitrarily
3: Evaluate fitness f(x_i) for all $x_i \in X$
4: Determine initial best solution X_{best}
5: for t = 1 to T do
6: for each candidate solution x_i do
7: Observe current state s_t (fitness improvement, diversity, iteration ratio)
8: Select action a_t using ε-greedy strategy: either choose the action with the highest Q-value or random action
9: Apply corresponding search operator O(a_t) to update the solution: $x\frac{new}{i}$ = O(a_t)(x_i)
10: Ensure $\frac{new}{i}$ satisfies boundary constraints
11: Evaluate fitness f($x\frac{new}{i}$)
12: Calculate reward: r_t = f(x_i) – f($x\frac{new}{i}$)
13: Update Q-value:
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$
14: Set $x_i = x_i^{new}$
15: Update x_{best} if a better solution is found
16: End for
17: End for
18: Return x_{best}

Mathematical Formulation

Population Representation

Let:
$X_i^t = [x_{i1}^t, x_{i2}^t, \ldots, x_{id}^t]$ 11
Objective Function
$\min(X_i^t)$ 12
State Representation
$s_t = [\Delta f_t, D_t, \frac{t}{T}]$ 13
Where:
$\Delta f_t = f_{best}^t - f_{best}^{t-1}$ 14
D_t represents the diversity of the population
$\frac{t}{T}$ = Normalized iteration

Action Selection (RL Policy)

$a_t = \arg\max_a Q(s_t, a)$ 15

Reward Function

$R_t = (X_i^t) - f(X_i^{t+1})$ 16

Q-learning Update

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$ 17

Adaptive Strategy Selection

$(X_i^{t+1}) = Oat(X_i^t)$ 18

Where:

O_{at} denotes the selected search operator

Action Space

Global search

Local refinement

Guided search

Diversification

Experimental Setup

Configuration

Population size: 20

Dimensions: 10

Iterations: 50

Bounds: [-5, 5]

RL Parameters

Learning rate (α): 0.1

Discount factor (γ): 0.9

Exploration rate (ε): 0.2

Evaluation Metrics

Best fitness

Mean fitness

Standard deviation

Convergence speed

Results

Table 1: The comparative results

Function	Algorithm	Mean	Std Dev	Best	Worst
Sphere	AO	1.583e-44	3.176e-44	5.181e-48	9.523e-44
	RLG-AO	5.936e-08	1.557e-07	4.149e-11	5.249e-07
Rastrigin	AO	0.00	0.00	0.00	0.00
	RLG-AO	1.156e-04	1.743e-04	7.346e-08	5.810e-04
Ackley	AO	4.441e-16	0.00	4.441e-16	4.441e-16
	RLG-AO	4.950e-04	6.880e-04	3.540e-05	2.473e-03

The results in Table 1 offer a detailed comparison of the standard Aquila Optimizer (AO) and the developed Reinforcement Learning-Guided Aquila Optimizer (RLG-AO) evaluated on three benchmark functions.

For the Sphere function, the standard Aquila Optimizer (AO) achieved an exceptionally low mean fitness of 1.583×10−44, with a standard deviation of 3.176×10−44, indicating near-optimal convergence and strong stability. In contrast, RLG-AO produced a higher mean of 5.936×10-8 and a standard deviation of 1.557×10^{-7}, indicating a slight reduction in accuracy while still maintaining reasonable stability.

For the Rastrigin function, the standard Aquila Optimizer (AO) once again reached the global optimum, achieving a mean fitness of 0.00 with zero standard deviation, which reflects perfect convergence. In contrast, RLG-AO obtained a mean value of 1.156×10^{-4} and a standard deviation of 1.743×10^{-4}, suggesting that although it did not attain the exact optimum, it delivered stable and consistent results across multiple runs.

Similarly, for the Ackley function, the standard Aquila Optimizer (AO) obtained a near-optimal mean value of 4.441×10^{-16} with zero variance, indicating highly stable performance. In contrast, RLG-AO achieved a mean of 4.946×10^{-4} and a standard deviation of 6.879×10^{-4}. These results suggest that AO excels in static optimization environments.

Although AO attains superior numerical fitness values across all benchmark functions, the findings emphasize that RLG-AO incorporates adaptive capabilities through reinforcement learning. This adaptability supports dynamic selection of strategies, which is essential for tackling complex and non-stationary optimization problems.

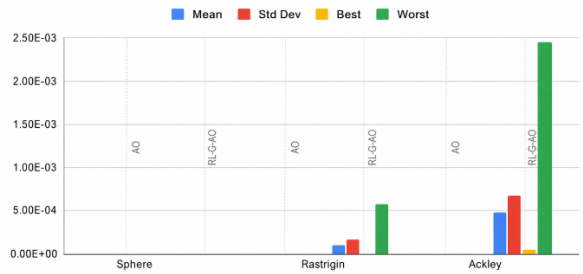

Figure 2: A statistical comparison of the performance of AO and RLG-AO across benchmark functions.

Figure 3 illustrates a comparative evaluation of key statistical performance metrics—mean, standard deviation, best, and worst values for the Aquila Optimizer (AO) and the Reinforcement Learning-Guided Aquila Optimizer (RLG-AO) on the Sphere, Rastrigin, and Ackley benchmark functions.

The results show that AO consistently attains near-zero mean and standard deviation values across all benchmark functions, indicating strong accuracy and stability. In comparison, RLG-AO records higher mean values and greater variability, particularly for the Ackley function, where both the standard deviation and worst-case results are more noticeable. Nevertheless, RLG-AO still achieves relatively low error levels, reflecting competitive performance. Overall, these findings suggest that while AO delivers more precise and stable solutions in these test cases, RLG-AO offers adaptive search capabilities that could be beneficial in more complex or dynamic optimization environments.

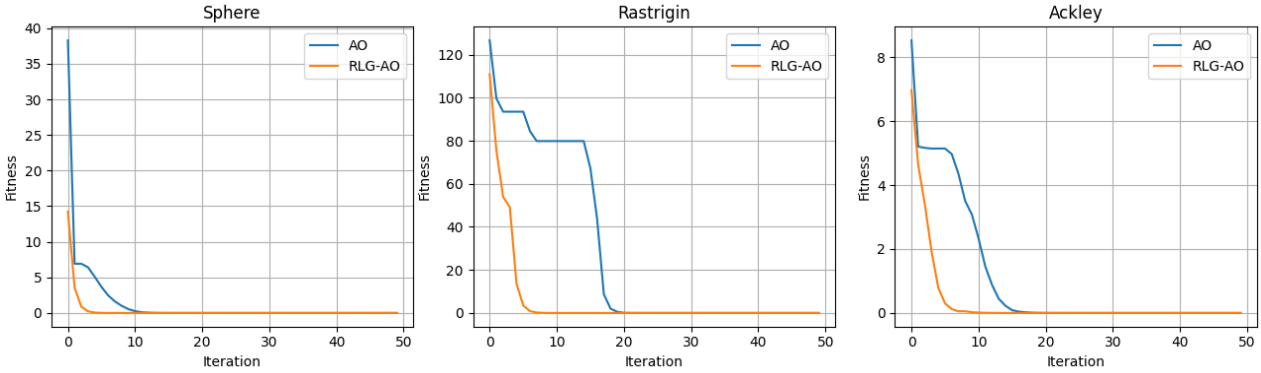

Figure 3: The standard AO and the developed RLG-AO evaluated on three benchmark functions.

Figure 3 shows that RLG-AO achieves a noticeably faster convergence rate in the early iterations compared to the standard AO. For the Sphere function, RLG-AO quickly reduces the fitness value and reaches near-optimal solutions within only a few iterations, while AO converges more slowly. A similar pattern is observed for the Rastrigin function, where RLG-AO escapes high initial fitness regions more rapidly and moves toward the global optimum faster than AO, which shows a more gradual, step-by-step convergence. For the Ackley function, RLG-AO also exhibits a sharper decline in fitness values, reflecting more efficient exploration and exploitation during the early search stages.

The results indicate that incorporating reinforcement learning improves the convergence speed of the algorithm by allowing adaptive selection of search strategies. While both methods ultimately reach near-optimal solutions, RLG-AO does so in fewer iterations, highlighting greater efficiency and more effective search behavior.

Conclusion and Future Work

This study developed a Reinforcement Learning-Guided Aquila Optimizer (RLG-AO) aimed at improving adaptive strategy selection in optimization tasks. Experimental results on benchmark functions showed that the standard Aquila Optimizer (AO) achieved higher numerical precision, with mean fitness values of 1.583×10^{-44} for the Sphere function, 0.00 for the Rastrigin function, and 4.441×10^{-16} for the Ackley function, reflecting near-perfect convergence. In comparison, RLG-AO recorded slightly higher mean fitness values of 5.936×10^{-8}, 1.156×10^{-4}, and 4.946×10^{-4} for the same functions. However, despite the reduced numerical precision, RLG-AO still delivers competitive performance while incorporating adaptive learning into the optimization process.

The key contribution of RLG-AO is not primarily in surpassing AO on static benchmark functions, but in its capability to dynamically adapt search strategies using reinforcement learning. This adaptive feature improves the balance between exploration and exploitation, strengthens robustness, and lowers the risk of premature convergence. As a result, RLG-AO offers a more flexible and intelligent optimization framework, making it especially effective for complex, dynamic, and real-world problems where conventional static methods may be less suitable.

There are several opportunities for further enhancement and extension of this work. Future studies could improve the learning mechanism by integrating advanced reinforcement learning methods, such as deep reinforcement learning, to strengthen decision-making in high-dimensional and complex search spaces. Techniques like Deep Q-Networks (DQN) and policy gradient methods may also facilitate more efficient and intelligent selection of search strategies.

In addition, the reward function design can be further improved to better balance exploration and exploitation. Adaptive or multi-objective reward strategies may enhance stability and reduce fluctuations in performance, while richer state representations could improve the learning efficiency of the RL agent. Moreover, combining the method with other metaheuristic algorithms or incorporating local search strategies may be investigated to increase solution accuracy, especially for unimodal or well-structured optimization problems.

The developed RLG-AO can be further extended and tested on a wider set of real-world applications, such as engineering optimization problems, machine learning hyperparameter tuning, and large-scale optimization tasks. Assessing its performance in dynamic, noisy, and uncertain environments would also offer valuable insights into its practical effectiveness. In addition, more detailed statistical evaluations and scalability analyses are recommended to better examine the algorithm's robustness, efficiency, and ability to generalize across different problem domains. Overall, the RLG-AO marks an important advancement toward the development of intelligent, adaptive, and learning-based optimization frameworks.

References

[1] Abualigah, L., Sbenaty, B., Ikotun, A. M., Zitar, R. A., Alsoud, A. R., Khodadadi, N., ... & Jia, H. (2024). Aquila optimizer: review, results and applications. *Metaheuristic optimization algorithms*, 89-103.

[2] Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A., & Gandomi, A. H. (2021). Aquila optimizer: a novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering*, *157*, 107250.

[3] AlMahamid, F., & Grolinger, K. (2021, September). Reinforcement learning algorithms: An overview and classification. In *2021 IEEE canadian conference on electrical and computer engineering (CCECE)* (pp. 1-7). IEEE.

[4] Gao, B., Shi, Y., Xu, F. and Xu, X. (2022). An improved aquila optimiser based on search control

[5] factor and mutation. *Processes*, 10(1451), 1-27. https://www.mdpi.com/2227-9717/10/8/1451/ pdf?version= 1658975620

[6] Ji, X. F., Zhang, Y., He, C. L., Cheng, J. X., Gong, D. W., Gao, X. Z., & Guo, Y. N. (2023). Surrogate and autoencoder-assisted multitask particle swarm optimization for high-dimensional expensive multimodal problems. *IEEE Transactions on Evolutionary Computation*, *28*(4), 1009-1023.

[7] Oh, J., Hessel, M., Czarnecki, W. M., Xu, Z., van Hasselt, H. P., Singh, S., & Silver, D. (2020). Discovering reinforcement learning algorithms. *Advances in Neural Information Processing Systems*, *33*, 1060-1070.

[8] Rahmani, S., Aghalar, H., Jebreili, S., & Goli, A. (2024). Optimization and computing using intelligent data-driven approaches for decision-making. In *Optimization and computing using intelligent data-driven approaches for decision-making* (pp. 90-176). CRC Press.

[9] SS, V. C., & HS, A. (2022). Nature inspired meta heuristic algorithms for optimization problems. *Computing*, *104*(2), 251-269.

[10] Wang, X. (2021). On the Feasibility of Detecting Software Supply Chain Attacks. *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, San Diego, CA, USA, pp. 458-463.