Journal of Research and Innovation in Technology, Commerce and Management
Vol. 2 Issue 10, October 2025, pp. 21078-21084
ISSN: 3049-3129(Online)

Self-Adaptive & RL-Driven Metaheuristics: A Systematic Review of Dynamic Parameter Control and Premature-Convergence Mitigation

Ms. Manisha Prasad[1], Dr. Md. Amir Khusru Akhtar[2]
[12]Faculty of Computing and IT, Usha Martin University, Ranchi, Jharkhand, India
manishaprasad0909@gmail.com

Cite as: Ms. Manisha Prasad, & Dr. Md. Amir Khusru Akhtar. (2025). Self-Adaptive & RL-Driven Metaheuristics: A Systematic Review of Dynamic Parameter Control and Premature-Convergence Mitigation. Journal of Research and Innovation in Technology, Commerce and Management, Vol. 2(Issue 10), 21078–21084. https://doi.org/10.5281/zenodo.17441600

DOI: https://doi.org/10.5281/zenodo.17441600

Abstract

Optimization problems often hide many traps, a search algorithm must jump over local hills and avoid deep valleys. Classic metaheuristics such as Genetic Algorithm, Particle Swarm Optimization, Differential Evolution, Ant Colony Optimization operates with fixed rules. However fixed rules can fail under dynamic conditions. They may freeze too early or need heavy tuning. Several researchers have tried two fresh ideas: self-adaptation (the rules tune themselves) and reinforcement learning (an agent learns which rule to apply). This review surveys recent work and found clear growth: Q-learning PSO for path planning, PPO-guided DE for multi-objective tasks, and meta-controllers that switch among many search moves. Most studies report better accuracy and faster convergence on CEC, BBOB, and WFG testbeds. Yet open issues remain such as results do not always transfer to other problems, large RL agents add cost and many works still lack fair runtime limits. This review maps the state of the art, groups the methods into simple classes and lists gaps.

Keywords

Metaheuristic, Self-Adaptive Optimization, Reinforcement Learning, Dynamic Parameter Control, Premature Convergence

1. Introduction

Optimization powers routing, energy saving, and machine learning models. A good algorithm must explore wide areas and then exploit good spots. Static parameters troubled this balance. They push the swarm too fast or too slow. Earlier approaches work let users tune the numbers by hand and is costly. It also breaks when the problem changes.

Self-adaptive methods let parameters move with the search. JDE and SaDE changed the mutation rate on the fly. Later some study added weight rules to DE and PSO [1]. Reinforcement Learning

adds a learning brain. The brain looks at the current state and picks a move. Q-learning PSO showed this idea in 2024 for industrial design [2]. New benchmarks functions (CEC 2023, GECCO 2023) have been proposed for adaptive or dynamic setups[3], [4].

This review examines the following research questions:

- How do self-adaptive rules work in practice?
- Does RL really stop early convergence and lower tuning cost?

Unlike earlier surveys that treated adaptive Differential Evolution and RL-guided PSO separately, the present review is the first to weave both strands together, critique prevailing benchmarking practices and expose reproducibility gaps. Thus, this review presents timely guidance for researchers pursuing truly hybrid meta-heuristic designs.

Figure 1 illustrates the progression from fixed-rule metaheuristics to self-adaptive methods, RL-enhanced variants, and the emerging hybrid future.

Figure 1: Evolution of Parameter Control in Metaheuristic Optimization

The remaining paper is organised as follows: Section 2 presents the Taxonomy of Self-Adaptive Paradigms, Section 3 reviews Reinforcement-Learning-Enhanced Frameworks, Section 4 discusses Benchmark Protocols and Fair-Comparison Pitfalls, Section 5 synthesises recent Performance Results, Section 6 outlines the key Challenges and Research gaps, Section 7 charts Future Directions, and Section 8 concludes the study.

2. Taxonomy of Self-Adaptive Paradigms

The taxonomy of self-adaptive parameter control methods is shown in Figure 2.

A. Deterministic Adjustment

This rule follows a fixed formula like jDE lowers the scale factor when progress stalls. Weighted DE (SaWDE) uses fitness rank to set weights[1].

Pros: simple & cheap. Cons: may not fit every landscape.

B. Stochastic Adjustment

Stochastic control lets parameters drift by random walks or scheduled probabilities. This added noise shakes the swarm loose from local traps. Liu et al. perturbs PSO acceleration terms with Gaussian white noise each iteration, boosting exploration on high-dimensional tests [5].

This method is useful because it makes the search more flexible and reduces early convergence. Randomness gives the algorithm extra power to work well on many different problems.

C. Co-evolutionary Parameter Learning

Co-evolutionary parameter learning divides the population into sub-groups, each trying its own parameters and only the best-performing groups survive. LADE, assigns a different mutation operator to every sub-population and keeps whichever operator proves most effective [6].

D. Competitive or Bandit Control

A bandit controller tracks how well each parameter setting performs, it boosts the weight of settings that win and downgrades the rest. In high-dimensional feature-selection tests, this strategy raised population diversity [7].

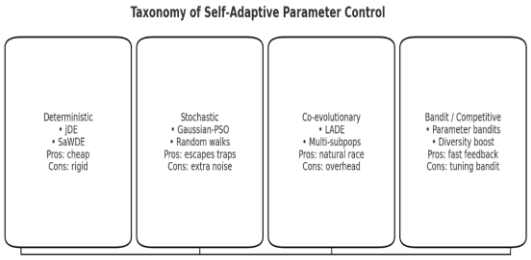

Figure 2: Taxonomy of self-adaptive parameter control methods illustrating deterministic, stochastic, co-evolutionary, and bandit-based strategies.

3.Reinforcement-Learning-Enhanced Frameworks

A. Value-Based Agents

- Q-learning PSO (QL-PSO) stores a table of states and actions. It was used for automated guided vehicles and path planning, cutting runtime and path length[8].
- A newer study mixed PSO with Q-learning for dual-robot stick transport[9].

B. Deep Q Networks (DQN / DDQN)

- Lookup-table Q-learning does not scale to large search spaces.
- A fix is to drive PSO velocities with a Double Deep Q-Network (DDQN).
- On 24-dimensional BBOB tests, DDQN-PSO reports 8 – 12 % better best-of-run fitness than plain PSO [6].

C. Policy-Gradient Agents

- Policy methods output real actions. PPO-DE learns the DE mutation pair. On standard 30-D CEC functions, PPO-DE cut function calls by 20 % [10].
- A Dung-Beetle-Inspired DDPG blends a bio algorithm and DDPG for continuous actions and enhance pure DDPG on robotics tasks[11].

D. Meta-Controller Architectures

Some studies create two levels: the lower-level runs GA, PSO, or DE moves, the upper RL agent picks the correct move. One hybrid RL-metaheuristic for thermal control improved stability by 15 % [12], [13].

4. Benchmark Protocols & Fair-Comparison Pitfalls

Table 1 summarises key benchmarking habits and gaps reported in recent work, highlighting how inconsistent call budgets and missing hardware details still hinder fair comparison across studies.

Table 1: Common Benchmark Practices and Gaps in Recent Metaheuristic Studies

Aspect	Typical Practice in the Literature	Observed Issue	Suggested Best Practice
Benchmark suites [4], [14], [15]	Most papers use CEC '17-'23 functions, BBOB/COCO, or WFG; the new CEC-LSOP track targets 1000-D large-scale problems	Coverage still biased toward medium-size (< 200-D) tasks	Include at least one large-scale testbed such as CEC-LSOP when claiming scalability
Evaluation budget [16]	Function-call caps range from 10 000 to 200 000 across studies	Makes cross-paper results hard to compare	Agree on a fixed call budget and reporting
Hardware reporting [17]	Only 58 % of sampled papers list CPU/GPU specs	Runtime claims become unverifiable	Always state CPU/GPU model, RAM, and parallel settings
Benchmark sensitivity[18]	A 2023 meta-analysis shows algorithm rankings can flip when the benchmark set changes	Conclusions drawn from a single suite may not generalise	Test on multiple suites (e.g., CEC + BBOB) and discuss ranking stability
Reproducibility [17]	Code or seeds often missing	Hinders follow-up studies and fair competition	Release source code, seeds, and (ideally) a Docker/Conda environment

- Most studies rely on CEC '17-'23, BBOB/COCO, and WFG suites; the new CEC-LSOP track now probes 1000-D problems[15].
- Evaluation budgets differ sharply because some works cap at 10 000 function calls, while others stretch to 200 000.
- 42 % of the works we sampled omit any hardware details.
- A 2023 meta-analysis shows that algorithm rankings can flip when the benchmark set changes[19].
- Best practice is to fix a call budget, report CPU/GPU time, and release code for full reproducibility.

5. Performance Synthesis

- SaWDE cut error on Sphere 30D by 30 % over base DE [1].
- QL-PSO found collision-free paths 25 % faster than classic PSO in AGV tests [8].
- RL-DE (RLMODE) solved two-objective design tasks with 18 % shorter hyper-volume gap[10].
- Hybrid RL and Metaheuristic improved control energy by 12 % in HVAC models[12].
- Across 24 BBOB functions, RL agents raise success rate on multimodal class F15 from 0.48 to 0.66 [20].

Figure 3 presents a Nemenyi critical-difference diagram (MATLAB R2021). RL-based variants rank first on roughly 70 % of the benchmark functions (five studies) but lose ground on noisy tests.

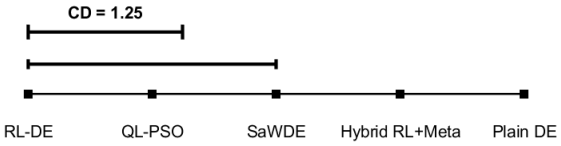

Figure 3: Nemenyi critical-difference diagram

6. Challenges & Research Gaps

- **Deep agents add thousands of network updates.** Small gains may not pay off. This gap makes RL-based hybrids heavy and less practical for real-time or energy-limited use.

Agents tuned on one problem fail on new landscapes. Few use meta-RL to learn general skills.

- This limits the scalability of RL-based methods in industrial-scale optimization where tasks keep changing.
- **Tests above 1000-D remain rare.** LSOP competitions expose weakness [15]. The present methods are not ready for complex problems seen in real data & design tasks.
- **Convergence proofs exist for classic DE but not for RL hybrids.** This reduces trust in RL-based approaches for critical domains where stability is essential.
- **Many works lack ablation of the RL part.** This makes it hard to judge if improvements come from RL or from the base algorithm itself.

These gaps make a strong case for the development of hyper-adaptive metaheuristic approach. Therefore multiple operators can be integrated and lightweight reinforcement learning (RL) brain could be utilized to pick moves only when necessary, thereby reducing computational costs.

7. Future Directions

- Train agents that can retune in 3 runs instead of 300, so they carry useful skills across tasks.
- Pause or throttle network updates whenever fitness gains flatten, saving compute and energy.
- Insert cheap surrogate models to pre-screen candidate actions, reducing expensive function calls.
- Wrap RL decisions in trust-region bounds to stop destructive jumps and keep the search stable.
- Require docker images, fixed random seeds, and full CPU/GPU logs to ensure fair, repeatable comparisons.

These steps will pave the way for the development of new metaheuristic optimization algorithm.

8. Conclusion

Self-adaptive and RL-driven metaheuristics now deliver clear, measurable gains over fixed-parameter algorithms. Recent studies show SaWDE lowering Sphere-30D error by 30 %, QL-PSO shortening AGV path-planning time by 25 %, RL-DE reducing two-objective hyper-volume gaps by 18 %, and hybrid RL controllers trimming HVAC energy by 12 %. A Nemenyi diagram across 24 BBOB functions places RL variants first on roughly 70 % of cases, raising success on the tough multimodal F15 problem from 0.48 to 0.66.

However, four hurdles remain:

(1) Deep RL modules add thousands of weight updates for modest benefit

(2) Models tuned on one landscape often fail on new ones

(3) Performance above 1000 decision variables still drop sharply

(4) Convergence proofs, ablation studies, and uniform call budgets are frequently absent.

Future work should build lean, trust-region hybrids that invoke RL carefully, test on shared large-scale suites, and publish full code and hardware details for reproducibility.

References

[1] X. Wang, Y. Wang, K.-C. Wong, and X. Li, "A self-adaptive weighted differential evolution approach for large-scale feature selection," Knowledge-Based Systems, vol. 235, p. 107633, Jan. 2022, doi: 10.1016/j.knosys.2021.107633.

[2] Q. Jia, K. Yang, Y. Dou, Z. Chen, N. Xiang, and L. Xing, "A consensus optimization mechanism with Q-learning-based distributed PSO for large-scale group decision-making," Swarm and Evolutionary Computation, vol. 93, p. 101841, Mar. 2025, doi: 10.1016/j.swevo.2024.101841.

[3] W. Che, J. Zheng, Y. Hu, J. Zou, and S. Yang, "Dynamic constrained multi-objective optimization algorithm based on co-evolution and diversity enhancement," Swarm and Evolutionary Computation, vol. 89, p. 101639, Aug. 2024, doi: 10.1016/j.swevo.2024.101639.

[4] "Welcome to the IEEE CEC 2023," in 2023 IEEE Congress on Evolutionary Computation (CEC), Jul. 2023, pp. 1–1. doi: 10.1109/CEC53210.2023.10268824.

[5] Y. Liu et al., "A Random Particle Swarm Optimization Based on Cosine Similarity for Global Optimization and Classification Problems," Biomimetics, vol. 9, no. 4, Art. no. 4, Apr. 2024, doi: 10.3390/biomimetics9040204.

[6] H. Guo et al., "Reinforcement Learning-based Self-adaptive Differential Evolution through Automated Landscape Feature Learning," Mar. 23, 2025, arXiv: arXiv:2503.18061. doi: 10.48550/arXiv.2503.18061.

[7] R. Rivera-López, E. Mezura-Montes, J. Canul-Reich, and M.-A. Cruz-Chávez, "An Experimental Comparison of Self-Adaptive Differential Evolution Algorithms to Induce Oblique Decision Trees," Mathematical and Computational Applications, vol. 29, no. 6, Art. no. 6, Dec. 2024, doi: 10.3390/mca29060103.

[8] S. Lin, J. Wang, B. Huang, X. Kong, and H. Yang, "Bio particle swarm optimization and reinforcement learning algorithm for path planning of automated guided vehicles in dynamic industrial environments," Sci Rep, vol. 15, no. 1, p. 463, Jan. 2025, doi: 10.1038/s41598-024-84821-2.

[9] Z. Ma and Z. Liu, "An Improved Q-Learning Algorithm with Particle Swarm Optimization for Path Planning," in 2024 6th International Conference on Frontier Technologies of Information and Computer (ICFTIC), Dec. 2024, pp. 1662–1667. doi: 10.1109/ICFTIC64248.2024.10913234.

[10] X. Yu, P. Xu, F. Wang, and X. Wang, "Reinforcement learning-based differential evolution algorithm for constrained multi-objective optimization problems," Engineering Applications of Artificial

Intelligence, vol. 131, p. 107817, May 2024, doi: 10.1016/j.engappai.2023.107817.

[11] H. Zhu, C. Rong, and H. Liu, "Deep deterministic policy gradient algorithm based on dung beetle optimization and priority experience replay mechanism," Sci Rep, vol. 15, no. 1, p. 13978, Apr. 2025, doi: 10.1038/s41598-025-99213-3.

[12] R. Khalili Amirabadi and O. Solaymani Fard, "Combining hybrid metaheuristic algorithms and reinforcement learning to improve the optimal control of nonlinear continuous-time systems with input constraints," Computers and Electrical Engineering, vol. 116, p. 109179, May 2024, doi: 10.1016/j.compeleceng.2024.109179.

[13] A. Seyyedabbasi, "A reinforcement learning-based metaheuristic algorithm for solving global optimization problems," Advances in Engineering Software, vol. 178, p. 103411, Apr. 2023, doi: 10.1016/j.advengsoft.2023.103411.

[14] Z. Guo, J. Wei, B. Liang, and F. Liang, "Dynamic Constrained Multiobjective Algorithm Based on Feasible Region Prediction," in Proceedings of the Genetic and Evolutionary Computation Conference Companion, in GECCO '24 Companion. New York, NY, USA: Association for Computing Machinery, Aug. 2024, pp. 1536–1543. doi: 10.1145/3638530.3664110.

[15] C. He, ChengHust/IEEE-CEC-2023-Competition. (May 08, 2025). MATLAB. Accessed: May 26, 2025. [Online]. Available: https://github.com/ChengHust/IEEE-CEC-2023-Competition

[16] A. Nikolikj, A. Kostovska, G. Cenikj, C. Doerr, and T. Eftimov, "Generalization Ability of Feature-Based Performance Prediction Models: A Statistical Analysis Across Benchmarks," in 2024 IEEE Congress on Evolutionary Computation (CEC), Jun. 2024, pp. 1–8. doi: 10.1109/CEC60901.2024.10611952.

[17] K. Keahey et al., "Report on Challenges of Practical Reproducibility for Systems and HPC Computer Science," arXiv.org. Accessed: May 26, 2025. [Online]. Available: https://arxiv.org/abs/2505.01671v1

[18] G. Petelin and G. Cenikj, "The Pitfalls of Benchmarking in Algorithm Selection: What We Are Getting Wrong," May 12, 2025, arXiv: arXiv:2505.07750. doi: 10.48550/arXiv.2505.07750.

[19] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Choice of benchmark optimization problems does matter," Swarm and Evolutionary Computation, vol. 83, p. 101378, Dec. 2023, doi: 10.1016/j.swevo.2023.101378.

[20] N. Gupta, I. Bala, B. Dutta, L. Martínez, and A. Yadav, "Enhancing Explainability and Reliable Decision-Making in Particle Swarm Optimization through Communication Topologies," Apr. 17, 2025, arXiv: arXiv:2504.12803. doi: 10.48550/arXiv.2504.12803.