

Decoupling Numerical and Structural Parameters: An Empirical Study on Adaptive Genetic Algorithms via Deep Reinforcement Learning for the Large-Scale TSP

Hongyu Wang

*School of Mathematics and Statistics
Xi'an Jiaotong University
Xi'an, China
3121670564@stu.xjtu.edu.cn*

Yuhan Jing

*School of Mechanical Engineering
Xi'an Jiaotong University
Xi'an, China
frankjing420@stu.xjtu.edu.cn*

Yibing Shi

*School of Mathematics and Statistics
Xi'an Jiaotong University
Xi'an, China
16121@stu.xjtu.edu.cn*

Enjin Zhou

*School of Mathematics and Statistics
Xi'an Jiaotong University
Xi'an, China
GINzej@stu.xjtu.edu.cn*

Haotian Zhang

*Frontier Institute of Science and Technology
Xi'an Jiaotong University
Xi'an, China
ht.zhang@xjtu.edu.cn*

Jialong Shi

*School of Mathematics and Statistics
Xi'an Jiaotong University
Xi'an, China
jialong.shi@xjtu.edu.cn*

Abstract—Proper parameter configuration is a prerequisite for the success of Evolutionary Algorithms (EAs). While various adaptive strategies have been proposed, it remains an open question whether all control dimensions contribute equally to algorithmic scalability. To investigate this, we categorize control variables into *numerical parameters* (e.g., crossover and mutation rates) and *structural parameters* (e.g., population size and operator switching), hypothesizing that they play distinct roles. This paper presents an empirical study utilizing a dual-level Deep Reinforcement Learning (DRL) framework to decouple and analyze the impact of these two dimensions on the Traveling Salesman Problem (TSP). We employ a Recurrent PPO agent to dynamically regulate these parameters, treating the DRL model as a probe to reveal evolutionary dynamics. Experimental results confirm the effectiveness of this approach: the learned policies outperform static baselines, reducing the optimality gap by approximately 45% on the largest tested instance (rl5915). Building on this validated framework, our ablation analysis reveals a fundamental insight: while numerical tuning offers local refinement, *structural plasticity* is the decisive factor in preventing stagnation and facilitating escape from local optima. These findings suggest that future automated algorithm design should prioritize dynamic structural reconfiguration over fine-grained probability adjustment. To facilitate reproducibility, the source code is available at <https://github.com/StarDream1314/DRLGA-TSP>

Index Terms—Hyper-heuristics, Genetic Algorithms, Deep Reinforcement Learning, Traveling Salesman Problem, Adaptive Parameter Control, Zero-Shot Generalization.

Corresponding author: Jialong Shi (jialong.shi@xjtu.edu.cn). This work was supported in part by the National Natural Science Foundation of China (Grants 12571590 and 12401670), the fellowship of China National Postdoctoral Program for Innovative Talents (Grant BX20240284), and the fellowship from the China Postdoctoral Science Foundation (Grant 2025M771731).

I. INTRODUCTION

Hyper-heuristics and Evolutionary Algorithms (EAs) are robust methodologies for solving complex NP-hard combinatorial optimization problems. However, their performance remains notoriously sensitive to parameter configurations [1], [2]. Traditionally, Adaptive Parameter Control (APC) relied on manually designed rules or fuzzy logic to adjust parameters online [3], [4]. As problem scales have grown, the field has shifted towards Automated Dynamic Algorithm Configuration (DAC), modeling parameter control as a sequential decision-making process [5], [6]. Deep Reinforcement Learning (DRL) has emerged as a leading approach in this domain, leveraging its ability to map high-dimensional optimization states to adaptive control actions [7], [8].

Despite DRL's promise, a critical gap persists in the literature. Most existing RL-for-EA frameworks focus almost exclusively on fine-tuning *numerical* parameters (e.g., crossover and mutation rates) within a static algorithmic scaffold [9], [10]. Alternatively, Neural Combinatorial Optimization (NCO) approaches use RL merely to initialize solutions before EA refinement, rather than controlling internal evolutionary dynamics [11]–[13]. These methods largely overlook *structural plasticity*—the dynamic adjustment of fundamental algorithmic components like population sizing and operator topologies. Theoretical analyses show that rigid structural configurations are fragile; slight deviations in problem characteristics can degrade algorithmic performance exponentially [14], [15]. A solver optimized for one instance often fails catastrophically on others because fine-tuning probabilities cannot overcome the bottleneck of a rigid population structure.

To address this rigidity, we propose a Dual-Level Deep Reinforcement Learning Genetic Algorithm (DRLGA). By treating structural parameters as first-class, learnable control variables alongside numerical ones, we formulate a unified closed-loop controller. We model the optimization as a Partially Observable Markov Decision Process (POMDP) and employ a **Recurrent Proximal Policy Optimization (Recurrent PPO)** agent equipped with Long Short-Term Memory (LSTM) to capture non-Markovian evolutionary histories. The agent acts as a high-level scheduler, regulating both numerical probabilities and structural modes, which are then executed by a high-performance evolutionary worker.

The overall framework of our proposed model is illustrated in Figure 1.

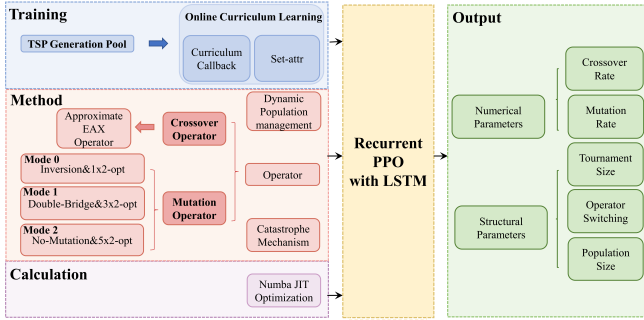


Fig. 1: A Dual-level Adaptive Control Architecture

Our contributions are summarized as follows:

- 1) **Dual-Level Control Framework:** We propose DRLGA, which synergizes dynamic population management and operator switching (structural) with traditional probability adjustments (numerical), significantly enhancing search versatility across varied fitness landscapes.
- 2) **Robust Zero-Shot Generalization:** Leveraging an LSTM-based Recurrent PPO scheduler trained solely on small-scale synthetic data (up to 1200 cities), our agent successfully generalizes zero-shot to massive benchmarks. It reduces the optimality gap by approximately 45% on the 5915-city instance (rl5915), achieving sustained stepwise descent where static methods stagnate.
- 3) **Decoupling Structural vs. Numerical Impact:** Through systematic ablation, we reveal that while numerical tuning aids local convergence, structural plasticity is the decisive factor in preventing stagnation in high-dimensional spaces, offering a new paradigm for automated algorithm design.

II. METHODOLOGY OVERVIEW

This paper formulates the adaptive control problem within Genetic Algorithms as a Partially Observable Markov Decision Process (POMDP). In this framework, the state S_t consists of a set of features describing the current runtime status of the GA. This includes 4 static features (problem size, aspect ratio, X distribution, Y distribution) and 11 dynamic features (time progress, optimality gap, recent improvement, stagnation

counter, population average gap, population variance, last action: CR, last action: MR, last action: Op Mode, current population size, coefficient of variation); the action A_t is a composite vector containing both numerical parameters (crossover rate, mutation rate) and structural parameters (population size, operator mode, tournament size); the optimization objective is to maximize the long-term improvement of solution quality within a given generation, which translates to minimizing the TSP path length.

Building upon these definitions, we construct a dual-level adaptive control architecture. The upper layer is the "Controller", where a Deep Reinforcement Learning agent observes the evolutionary state and outputs high-level control strategies. The lower layer is the "Worker", where a high-performance evolutionary algorithm executes specific search tasks using the specified parameters and feeds back the execution results. Information flows cyclically between the upper and lower layers: the Controller generates action A_t based on the current state S_t , and the Worker executes A_t , causing the state to evolve to S_{t+1} and generating a reward R_t . The specific implementation process is depicted in Figure 2. Its design adheres to the principles of decoupling and structural plasticity, ensuring that the agent can achieve macro-control over the search process without interfering with the underlying search efficiency.

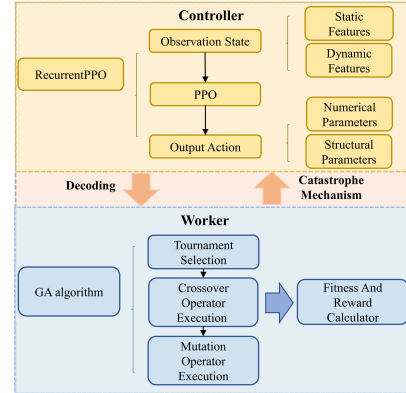


Fig. 2: The Upper Controller and The Lower Worker

III. DETAILED IMPLEMENTATION

A. The Neural Controller

To address the non-Markovian characteristics of the evolutionary search process—where the optimal action choice depends not only on the current state but also intimately on historical improvement trends and stagnation information—we adopt the Recurrent Proximal Policy Optimization (Recurrent PPO) algorithm, integrated with Long Short-Term Memory (LSTM) networks, as the high-level controller.

For state representation and memory, the controller accepts search states as time-series inputs. Unlike standard feed-forward networks, the LSTM extracts cross-timestep evolutionary features by maintaining a memory of the entire trajectory. Let

o_t denote the observation vector (feature state) at generation t . The LSTM updates its hidden state h_t and cell state c_t as follows:

$$(h_t, c_t) = \text{LSTM}(o_t, h_{t-1}, c_{t-1}; \theta_{enc}) \quad (1)$$

Here, h_t serves as a compact representation (or belief state) of the entire history $\tau_t = \{o_0, \dots, o_t\}$, effectively converting the non-Markovian observation sequence into a Markovian latent state suitable for subsequent decision-making.

For policy generation and action mapping, the policy network π_θ outputs a continuous action vector based on the history embedding h_t . We model the stochastic policy as a multivariate Gaussian distribution:

$$a_t \sim \pi_\theta(\cdot|h_t) = \mathcal{N}(\mu_\theta(h_t), \Sigma_\theta(h_t)) \quad (2)$$

where μ_θ and Σ_θ are neural networks mapping h_t to the mean and (co)variance of the action space, respectively. The sampled raw action a_t is then mapped to specific GA parameters. Certain action dimensions are discretized for structural parameters (e.g., Operator Modes 0/1/2 and Tournament Size TS), while the remaining dimensions are directly mapped to continuous numerical parameters (e.g., Crossover Rate CR , Mutation Rate MR).

Regarding the optimization objective, this memory-equipped decision-making mechanism enables the controller to perceive long-term dynamic information. To train the controller, we maximize the PPO objective function, which incorporates a clipping mechanism to ensure training stability:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right] \quad (3)$$

where the probability ratio $r_t(\theta)$ is explicitly conditioned on the historical context maintained by the LSTM:

$$r_t(\theta) = \frac{\pi_\theta(a_t|h_t)}{\pi_{\theta_{old}}(a_t|h_t^{old})} \quad (4)$$

By optimizing this objective, the controller learns to perform multi-stage global switching and parameter adaptation based on the inferred search dynamics.

B. The High-Performance Evolutionary Worker

This module constitutes a high-performance Genetic Algorithm, accelerated using Numba JIT, responsible for executing computationally intensive combinatorial optimization tasks. Specifically, it incorporates a dual-level adaptive mechanism: at the numerical level, the algorithm dynamically adjusts operator intensity and selection pressure based on the parameters output by the controller; at the structural level, it integrates dynamic population management and multi-mode operator switching to adapt to the characteristics of various periods in the evolutionary process.

Dynamic Population Management enables the algorithm to flexibly adjust the population size according to the current evolutionary stage and performance. A smaller population is utilized in the early stages to improve computational efficiency and convergence speed, while the population size is gradually

expanded in later stages or when improvement slows, to enhance the exploration capability of the search space.

Operator Switching defines three distinct structural modes to handle different evolutionary stages:

- **Mode 0 (Initial Exploration/Stationary Stage):** Applicable to the early search phase. It primarily combines the Inversion operator with one 2-opt operation to rapidly improve path quality while maintaining solution diversity.
- **Mode 1 (Intensified Exploration Stage):** Applicable when the search approaches a local optimum or enters a stagnation phase. It employs the Double-Bridge operator combined with three 2-Opt operations to increase path perturbation capability, thereby facilitating escape from local optima.
- **Mode 2 (Fine-grained Convergence Stage):** Activated in the late convergence phase. It disables random mutation and focuses on five 2-Opt local optimization operations to achieve fine-grained adjustment of the path, further approximating the optimal solution.

Their specific working principles are illustrated in Figure 3.

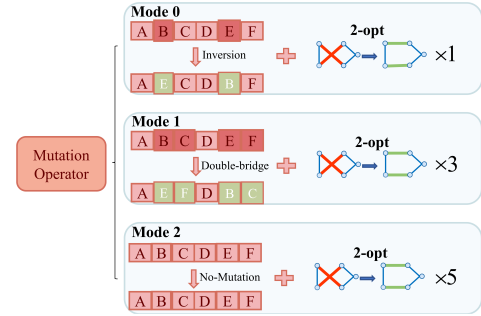


Fig. 3: Mutation Operator

For the crossover operator, we employ an approximate EAX operator, whose working principle is illustrated in Figure 4.

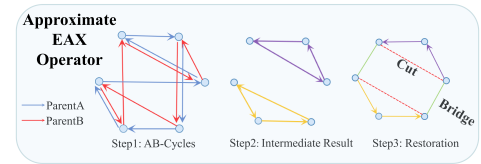


Fig. 4: Crossover Operator

Furthermore, when the evolutionary search yields no improvement for a prolonged period (e.g., every hundred generations) and conventional mode switching fails to significantly boost performance, a catastrophic restart mechanism is judiciously introduced to reset the solution space, thereby restoring diversity and search vitality. As a supplementary design to enhance robustness, this catastrophic restart mechanism is implemented independently of the staged operator modes.

C. Training Strategy

To train a control policy with robust generalization capabilities, we adopt an online curriculum learning scheme, which

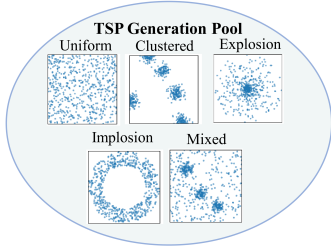


Fig. 5: Examples of TSP Instances Generated by the TSP Generator Pool

effectively mitigates the cold-start problem in reinforcement learning. The entire training process is divided into four progressively advancing stages: starting from simple instances with 50–150 cities, gradually increasing to medium-scale problems with 400–800 cities, and finally encompassing large-scale problems with up to 1200 cities. To enhance the model’s adaptability to diverse topological structures, a TSP Generator Pool was specifically constructed, as illustrated in Figure 5. This pool can generate synthetic examples with various spatial distributions, such as uniform, clustered, explosion, implosion, and mixed types, thereby compelling the controller to learn general switching strategies independent of specific structures.

In terms of engineering implementation, the combination of CurriculumCallback and set_attr mechanisms ensures the synchronization of curriculum progress in a multi-process environment, thereby guaranteeing the stability and efficiency of distributed training. During the inference phase, the LSTM hidden states are continuously passed between steps to maintain contextual information. The controller then operates in a deterministic manner on test samples, achieving plug-and-play zero-shot generalization capabilities without any online fine-tuning.

IV. EXPERIMENTS AND ANALYSIS

In this section, our experiments are divided into two main parts: the first compares the performance of our DRLGA with a Static GA, while the second is an ablation study investigating whether numerical or structural parameters have a more significant impact on GA performance.

Problem Setting For performance comparison, we selected large-scale instances: fl1577, u2152, pcb3038, fnl4461, and rl5915 from **TSPLIB** to test the performance of DRLGA against a static GA. For the ablation study, we selected small-scale (kroA100), medium-scale (lin318, pcb442), and large-scale (rat783, pr1002) instances from **TSPLIB**.

Training. The proposed DRLGA model is trained on TSP instances ranging from 50 to 1200 cities, sampled from a custom TSP Generator Pool comprising diverse topologies to ensure training stability. Optimization is performed using Adam with an initial learning rate of 3×10^{-4} , adhering to a linear decay schedule based on training progress. Leveraging the Recurrent PPO architecture with a batch size of 2048, the model trains across 64 parallel environments for 5 million timesteps. This computationally intensive process was executed

on a high-performance node equipped with a 64-core AMD EPYC 7452 CPU (@2.35GHz), DDR4 memory, and InfiniBand high-speed interconnects to maximize parallel efficiency.

Baselines We compared the DRLGA against a highly optimized Static GA baseline, which utilizes the exact same Numba-accelerated kernel but operates with fixed, empirically tuned parameters (CR=0.8, MR=0.2, Population Size=500, Approximate EAX crossover operator + Mode 0, with Mode 2 activated every 50 generations).

Metrics We report the optimality gap, inference time, and dynamic parameter trajectories (for both structural and numerical parameters) in Experiment A to analyze the agent’s decision-making process. Experiment B validates the method’s robustness through average convergence trends and statistical box plots. The optimality gap is calculated as $\frac{L_{pred} - L_{opt}}{L_{opt}} \times 100\%$, serving as the standard measure of approximation quality. Each experiment was repeated 30 times independently with different random seeds to mitigate the impact of randomness. We report the mean metrics to evaluate the general performance. For experiment data from all test sets mentioned in this experiment, please refer to <https://github.com/StarDream1314/DRLGA-TSP>.

A. Performance Comparison: DRLGA vs. Static GA

As shown in Table I, DRLGA demonstrates superior asymptotic precision across all instances. We acknowledge that DRLGA incurs a higher computational overhead, with inference times averaging approximately two times that of the Static GA. However, it is crucial to emphasize that the primary objective of this study is not to engineer a runtime-optimized commercial solver, but rather to validate the feasibility of the DRL-driven hyper-heuristic framework and to investigate the distinct roles of structural versus numerical parameters. The significant reduction in the optimality gap serves as a proof of concept, demonstrating that the learned adaptive policies can unlock performance gains that static configurations cannot achieve, prioritizing scientific insight into evolutionary dynamics over raw computational speed.

TABLE I: Performance Comparison

Instance	Optimality Gap (%)		Inference Time (s)	
	Static GA	DRLGA (Ours)	Static GA	DRLGA (Ours)
fl1577	9.10	4.44	≈183	≈347
u2152	13.50	4.57	≈426	≈869
pcb3038	11.53	4.89	≈746	≈1606
fnl4461	9.84	6.60	≈1,807	≈3903
rl5915	19.19	10.48	≈2,821	≈6,240

As presented in Figure 6, the static GA exhibits typical early stagnation (an "L-shaped" convergence curve), failing to escape local optima. In contrast, DRLGA maintains a sustained "step-wise" descent in its optimality gap. Notably, on u2152, DRLGA reduces the optimality gap from 13.50% (Static GA) to 4.57%; for the largest instance, rl5915, the gap drops from

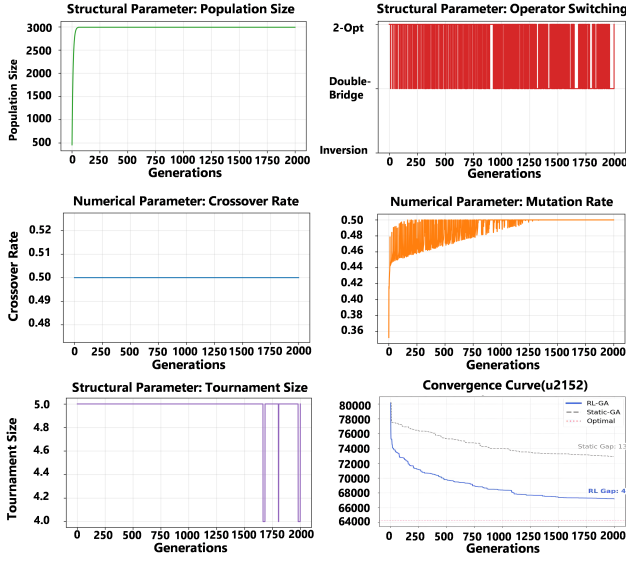


Fig. 6: Performance Comparison

19.19% to 10.48%. This confirms the agent’s superior ability to navigate complex fitness landscapes where static parameters prove inadequate.

The parameter trajectories reveal three intelligent mechanisms driving this superior performance:

1. **Resource Saturation:** The agent maximizes population size early in the search process to ensure sufficient information-carrying capacity for large-scale problem exploration.

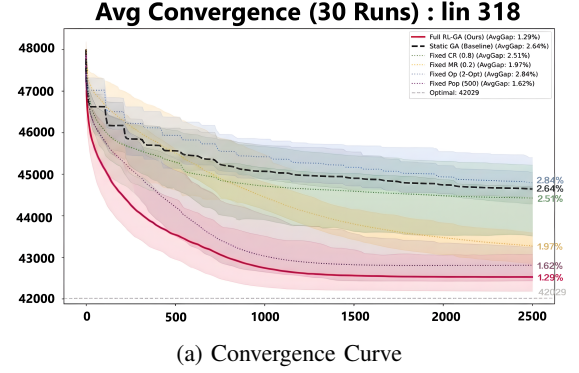
2. **Hyper-heuristic Oscillation:** The agent frequently switches between operators (Mode 1 and Mode 2), leveraging topological complementarity to disrupt population homogeneity and prevent premature convergence.

3. **Inverse Annealing:** This refers to a counter-intuitive behavior where mutation rates increase towards the end of evolution (approaching 0.5). We interpret this as an adaptive response to the late-stage “diversity crisis,” forcefully injecting entropy to break stagnation in large-scale instances.

It is noteworthy that DRLGA achieves robust zero-shot generalization, embodying the principle of “Train Small, Solve Large.” Trained solely on synthetic datasets with $N \in [50, 1200]$, the agent performs zero-shot inference on TSPLIB instances with up to $N = 5915$ cities. Despite a five-fold increase in problem size, the DRLGA agent avoids the “Out-of-Distribution” failure common in constructive neural heuristics. On rl5915, it achieves an approximately 50% improvement over the baseline, demonstrating that the learned control logic is robust against variations in problem dimensionality. By observing normalized evolutionary statistics rather than raw node coordinates, the state representation becomes scale-invariant. Consequently, the learned meta-control policies (e.g., “Inverse Annealing” during stagnation) function as universal optimization laws, applicable regardless of the problem size.

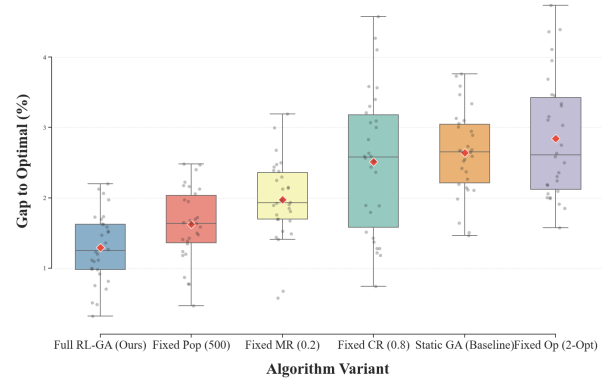
B. Ablation Study: Numerical Parameters vs. Structural Parameters

To disentangle the individual impacts of distinct control variables, we performed a systematic ablation study by classifying hyperparameters into structural (population size, operator mode) and numerical (crossover, mutation rates) categories. The results are partly shown in Figure 7.



(a) Convergence Curve

Performance Comparison on lin318



(b) Box Plot

Fig. 7: Analysis of lin318 Instance

As detailed in Table II, a distinct hierarchy emerges from the empirical data: structural adaptability functions as the dominant factor for scalability, whereas numerical parameters occupy a secondary role primarily focused on fine-tuning. Performance degradation was most acute when the algorithm was confined to a static operator (“Fixed Op” setting), particularly on large-scale instances. This evidence confirms that dependence on a singular neighborhood topology (e.g., 2-Opt) invariably results in stagnation at local optima. A similar trend was observed with fixed population sizes (“Fixed Pop” setting), highlighting a deficiency in information-carrying capacity. This validates the necessity of dynamic population scaling to sustain genetic diversity and exploration potential in high-dimensional search spaces.

Conversely, constraints on numerical parameters produced a more moderate impact. While the “Fixed CR” variant displayed sensitivity on medium-sized problems like lin318

TABLE II: Ablation Study

Dataset	Experiment	Gap (%)			
		Mean	Std	Min	Max
kroA100	Fixed CR (0.8)	0.07	0.02	0.06	0.12
	Fixed MR (0.2)	0.08	0.03	0.06	0.12
	Fixed Op (2-Opt)	0.18	0.06	0.12	0.28
	Fixed Pop (500)	0.08	0.03	0.06	0.12
	Full DRLGA (Ours)	0.08	0.03	0.06	0.12
	Static GA (Baseline)	0.18	0.06	0.12	0.28
lin318	Fixed CR (0.8)	2.39	0.96	0.74	4.58
	Fixed MR (0.2)	1.93	0.69	0.58	3.19
	Fixed Op (2-Opt)	2.90	0.79	1.57	4.73
	Fixed Pop (500)	1.71	0.54	0.47	2.48
	Full DRLGA (Ours)	1.35	0.50	0.32	2.20
	Static GA (Baseline)	2.58	0.63	1.46	3.76
pcb442	Fixed CR (0.8)	1.19	0.47	0.47	2.80
	Fixed MR (0.2)	1.29	0.52	0.52	2.31
	Fixed Op (2-Opt)	3.69	0.78	2.04	5.23
	Fixed Pop (500)	1.19	0.50	0.34	2.35
	Full DRLGA (Ours)	1.02	0.44	0.35	2.28
	Static GA (Baseline)	2.65	0.82	0.77	4.51
pr1002	Fixed CR (0.8)	3.80	0.79	2.47	6.03
	Fixed MR (0.2)	5.04	0.65	3.32	6.40
	Fixed Op (2-Opt)	6.36	0.88	4.88	8.40
	Fixed Pop (500)	4.38	0.72	2.83	6.00
	Full DRLGA (Ours)	3.85	0.80	2.68	5.90
	Static GA (Baseline)	5.16	0.77	3.66	6.82

(2.39% gap), the "Fixed MR" variant showed unexpected robustness on the massive pr1002 instance. By maintaining a gap of 5.04%—significantly outperforming the 6.36% gap of the structural failure ("Fixed Op" setting)—the results suggest a critical distinction: while mutation intensity regulates convergence velocity, it is less prone to inducing irreversible stagnation than structural rigidity. The search process may proceed more slowly with static mutation, but it can be effectively halted by an inflexible topology. Ultimately, the Full DRLGA secured the lowest mean gap across all datasets (e.g., 1.02% on pcb442). This superiority stems from a synergistic orchestration: the agent utilizes structural shifts to break out of basins of attraction, while reserving numerical adjustments to accelerate exploitation within promising regions.

V. CONCLUSION AND FUTURE WORK

This paper proposed a dual-level adaptive Genetic Algorithm framework, reinforced by Recurrent PPO, which decouples the control of numerical and structural parameters for fine-grained management of the evolutionary search process. Through systematic experimentation and analysis, we have not only validated the efficiency of this framework in solving large-scale Traveling Salesman Problems but also derived a critical insight that challenges traditional perspectives in evolutionary computation: when scaling to complex, high-dimensional

optimization tasks, adaptive structural management (such as dynamic population size and operator switching) has a far more decisive impact on performance than traditional numerical probability fine-tuning. This work establishes a new paradigm for designing scalable, intelligent optimization systems by elevating adaptation from solely the numerical level to encompass the structural level. Future work will extend this framework to constrained domains like the Vehicle Routing Problem (VRP) to test the transferability of learned meta-policies. Additionally, we aim to integrate automatically generated operators and apply this control mechanism to other evolutionary algorithms, such as Differential Evolution, to further generalize the findings on structural plasticity.

REFERENCES

- [1] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, Dec. 2013.
- [2] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, Apr. 2020.
- [3] D. Farinati and L. Vanneschi, "A survey on dynamic populations in bio-inspired algorithms," *Genet. Program. Evolvable Mach.*, vol. 25, no. 2, Dec. 2024, Art. no. 19.
- [4] B. Subburaj and S. Miruna Joe Amali, "A fuzzy system based self-adaptive memetic algorithm using population diversity control for evolutionary multi-objective optimization," *Sci. Rep.*, vol. 15, no. 1, Feb. 2025, Art. no. 5735.
- [5] S. Adriaensens, A. Biedenkapp, G. Shala, N. Awad, T. Eimer, M. Lindauer, and F. Hutter, "Automated dynamic algorithm configuration," *J. Artif. Intell. Res.*, vol. 75, pp. 1633–1699, Dec. 2022.
- [6] E.-G. Talbi, "Machine learning into metaheuristics: A survey and taxonomy," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–32, Jul. 2022.
- [7] G. Shala, A. Biedenkapp, N. Awad, S. Adriaensens, M. Lindauer, and F. Hutter, "Learning step-size adaptation in CMA-ES," in *Proc. 16th Int. Conf. Parallel Probl. Solving Nat.* Leiden, Netherlands: Springer, 2020, pp. 691–706.
- [8] F. Berto, C. Hua, J. Park, L. Luttmann, Y. Ma, F. Bu, J. Wang, H. Ye, M. Kim, S. Choi, N. G. Zepeda, A. Hottung, J. Zhou, J. Bi, Y. Hu, F. Liu, H. Kim, J. Son, H. Kim, D. Angioni, W. Kool, Z. Cao, Q. Zhang, J. Kim, J. Zhang, K. Shin, C. Wu, S. Ahn, G. Song, C. Kwon, K. Tierney, L. Xie, and J. Park, "RL4CO: An extensive reinforcement learning for combinatorial optimization benchmark," in *Proc. 31st ACM SIGKDD Conf. Knowl. Discov. Data Min.*, vol. 2. Toronto, ON, Canada: ACM, Aug. 2025, pp. 5278–5289.
- [9] A. Woodcock and L. Zhang, "Genetic algorithm with reinforcement learning based parameter optimisation," in *Proc. 2024 IEEE Int. Conf. Syst. Man Cybern.* Kuching, Malaysia: IEEE, Oct. 2024, pp. 4833–4839.
- [10] T. Zhou, W. Zhang, B. Niu, P. He, and G. Yue, "Parameter control framework for multiobjective evolutionary computation based on deep reinforcement learning," *Int. J. Intell. Syst.*, vol. 2024, Jan. 2024, Art. no. 6740701.
- [11] Y. Ruan, W. Cai, and J. Wang, "Combining reinforcement learning algorithm and genetic algorithm to solve the traveling salesman problem," *J. Eng.*, vol. 2024, no. 6, Jun. 2024, Art. no. e12393.
- [12] I. Bello, H. Pham, Q. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proc. 5th Int. Conf. Learn. Represent.* Toulon, France: OpenReview, Feb. 2017.
- [13] P. Li, J. Hao, H. Tang, X. Fu, Y. Zheng, and K. Tang, "Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms," *IEEE Trans. Evol. Comput.*, vol. 29, no. 5, pp. 1707–1728, Oct. 2025.
- [14] M. Kaufmann, M. Larcher, J. Lengler, and X. Zou, "Self-adjusting population sizes for the (1,λ)-EA on monotone functions," *Theor. Comput. Sci.*, vol. 979, Nov. 2023, Art. no. 114181.
- [15] L. Deng and S. Liu, "Unlocking new potentials in evolutionary computation with complex network insights: A brief survey," *Arch. Comput. Methods Eng.*, Jun. 2025.