DYNAMIC ALGORITHM CONFIGURATION WITH EVOLUTION-AWARE STATE REPRESENTATION FOR MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, deep reinforcement learning-based dynamic algorithm configuration (DRL-DAC) has been widely studied in evolutionary computation. However, existing DRL-DAC methods mainly focus on the structure of the current population while neglecting inter-generational evolutionary dynamics, leading to an insufficient characterization of the evolutionary state. To address this limitation, we propose an evolution-aware state representation that incorporates the Kullback–Leibler (KL) divergence between consecutive populations to capture inter-generational transitions. Combined with search-stage information, this metric guides the feature-wise reweighting of node embeddings, providing the RL agent with a more discriminative state representation for accurate parameter adaptation. Extensive experiments on representative multi-objective combinatorial optimization problems demonstrate that our method significantly improves overall optimization performance, achieving superior convergence and diversity compared to state-of-the-art baselines. Furthermore, evaluations under large-scale and many-objective settings highlight its strong generalization capabilities. The source code is available at `https://github.com/AetherrehteA/DAC`.

1 INTRODUCTION

Multi-objective combinatorial optimization problems (MOCOPs) arise widely in practical applications such as vehicle routing Zajac & Huber (2021), production scheduling Zhang et al. (2024), and resource allocation Vergara et al. (2023). Owing to the discrete nature of their decision spaces and conflicting objectives, exact methods struggle to obtain the complete Pareto-optimal set within a limited time budget. Multi-objective evolutionary algorithms (MOEAs) address this by maintaining a population of candidate solutions in a single run, making them primary methodologies for MOCOPs Deb et al. (2002); Zhou et al. (2011). However, MOEA performance is highly sensitive to hyperparameter configurations. While early search stages necessitate population diversity for sufficient space exploration, later stages demand a shift toward convergence to drive non-dominated solutions toward the Pareto front. Consequently, static parameter settings fail to adapt to these dynamic search requirements Gomes Pereira de Lacerda et al. (2021).

Reinforcement learning (RL) is promising for learning search and decision policies in combinatorial optimization via environment interaction Mazyavkina et al. (2021). In evolutionary computation, RL is widely applied to dynamic algorithm configuration (DAC), operator selection, solution generation, and search-strategy control Song et al. (2024). Specifically, DAC provides an effective framework for adaptive EA control by dynamically adjusting parameters during the optimization process Biedenkapp et al. (2020). In RL-based DAC, DAC is modeled as a sequential decision-making problem, in which the agent selects parameters based on the current state, the EA executes the corresponding operation, and a reward reflects improvement in population quality.

In this process, the state representation dictates which aspects of the search process are observable to the agent. If the state fails to sufficiently characterize the current population and its trajectory, the agent cannot reliably select parameter configurations aligned with current search requirements. Consequently, constructing an informative search-state representation remains a central challenge in RL-based DAC.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Existing studies characterize evolutionary search states through several representative strategies. MADAC constructs states from handcrafted statistical indicators and their historical variations, thereby explicitly incorporating search-history information but relying on manually designed features Xue et al. (2022). GS-MODAC instead represents the current population as an objective-space graph and learns a population-level embedding using a graph neural network Reijnen et al. (2025b). More recently, STN-MODAC constructs a Search Trajectory Network that retains solution transitions across generations and employs a temporal graph model to encode structural and temporal search dynamics Reijnen et al. (2025a).

While GS-MODAC effectively captures the structure of the current population, it does not explicitly describe how the population has changed from the preceding generation. Thus, populations at the same search stage may exhibit similar objective-space structures while undergoing different degrees of inter-generational change. STN-MODAC addresses this issue through explicit multi-generation solution trajectories. In contrast, our method summarizes the distributional change between two consecutive populations using KL divergence, providing a low-dimensional population-level description of recent change without maintaining a historical trajectory graph.

Motivated by this perspective, we propose an evolution-aware state representation for the dynamic configuration of multi-objective evolutionary algorithms. Building upon the current population graph, the method combines the KL divergence between consecutive population distributions Zhang et al. (2023) with normalized search progress to construct an evolution-aware feature vector. This feature adaptively modulates the latent dimensions of the graph-based population representation, and the modulated representation is concatenated with the evolution-aware feature vector to construct the policy state. Consequently, the state captures both the current population structure and recent inter-generational distributional change, enabling adaptive parameter configuration throughout the evolutionary process. The main contributions of this work are summarized as follows:

1. We propose an evolution-aware population representation integrating the graph structure of the current population with inter-generational distributional changes, enabling the reinforcement learning agent to capture both spatial structures and evolutionary dynamics.
2. We design an evolution-guided feature modulation mechanism that derives feature-wise weights from normalized search progress and consecutive-population KL divergence to adaptively modulate the latent population representation.
3. We conduct extensive experiments on multi-objective vehicle routing and flexible job shop scheduling problems, validating the proposed method through performance comparisons, generalization tests, and comprehensive ablation studies.

2 Background and Related Work

2.1 Multi-objective Combinatorial Optimization Problem

Multi-objective combinatorial optimization aims to simultaneously optimize multiple conflicting objectives over a discrete feasible solution space. Let $\mathcal{X}$ denote the set of feasible solutions. A problem with M minimized objectives can be mathematically formulated as

$$\min_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x}))\,,$$

where $f_m(\boldsymbol{x})$ denotes the m-th objective function to be minimized. For two feasible solutions $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$, $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_2$ ($\boldsymbol{x}_1 \succ \boldsymbol{x}_2$) if $\boldsymbol{x}_1$ is no worse than $\boldsymbol{x}_2$ across all objectives and strictly superior in at least one. Solutions not dominated by any other feasible solution are called Pareto-optimal solutions, and their mapping in the objective space forms the Pareto front. Due to conflicting objectives, the primary goal of multi-objective combinatorial optimization is to approximate the Pareto front with a set of non-dominated solutions that achieve strong convergence and uniform diversity. MOEAs derive this approximate Pareto front through population-based search, non-dominated sorting, and diversity preservation, serving as a primary approach for complex problems such as vehicle routing, production scheduling, and resource allocation.

2.2 Algorithm Configuration

Algorithm configuration aims to identify parameter combinations from a predefined configuration space that improve the performance of a target algorithm. Traditional automatic configuration methods typically search for parameter settings offline before execution Schede et al. (2023). Representative methods include ParamILS based on iterative local search Hutter et al. (2009), irace using a racing mechanism to eliminate poorly performing configurations López-Ibáñez et al. (2016), and SMAC3 employing surrogate models to guide configuration search Lindauer et al. (2022). Although these methods reduce manual trial-and-error, the resulting configurations remain fixed during execution.

Dynamic algorithm configuration (DAC) instead adapts parameters online according to runtime feedback Biedenkapp et al. (2020); Adriaensen et al. (2022). Biedenkapp et al. cast DAC as a contextual Markov decision process, where an RL policy learns to map search states to parameter configurations via interaction Biedenkapp et al. (2020). For heterogeneous parameter control, MADAC adopts a multi-agent framework that assigns independent agents to different parameters and constructs states from handcrafted statistical features and their historical variations Xue et al. (2022). While this incorporates search-history information, it relies on manually designed statistics, window sizes, and normalization schemes.

To reduce reliance on handcrafted features, GS-MODAC represents the current population as an objective-space graph and uses a graph neural network to learn population-level representations Reijnen et al. (2025b). Its state mainly captures the current population structure and normalized search progress, without explicitly describing changes from the preceding generation. More recently, STN-MODAC models multi-generation solution transitions through a Search Trajectory Network and a temporal graph model Reijnen et al. (2025a), explicitly retaining historical solution trajectories across generations. These studies suggest that incorporating temporal information can enrich the representation of evolutionary search states beyond a single population snapshot. At the same time, they motivate exploring more compact mechanisms that capture recent evolutionary dynamics without explicitly maintaining multi-generation trajectories. In contrast to such explicit trajectory modeling, our method summarizes the distributional change between two consecutive populations using KL divergence and integrates this low-dimensional population-level summary with the current population graph.

3 Method

3.1 Overall framework

To enable the reinforcement learning agent to distinguish populations with similar spatial structures but different evolutionary dynamics, we propose an evolution-aware state representation method for DAC. Figure 1 illustrates the overall interaction loop between the evolutionary algorithm and the reinforcement learning agent. At each generation, the current population is converted into a Pareto-dominance graph and encoded via a graph neural network. Concurrently, the distributional shift between the current and preceding populations is quantified and combined with the normalized search progress to form an evolution-aware feature vector. This vector is mapped to feature-wise weights that modulate the latent population representation according to the search stage and recent inter-generational change. The resulting graph representation is then concatenated with the evolution-aware feature vector to construct the state input for the reinforcement learning agent. Based on this comprehensive state, the agent determines the crossover and mutation probabilities for the current generation. The evolutionary algorithm subsequently performs reproduction, evaluation, and environmental selection to yield the next population, utilizing the resulting improvement in population quality as reward feedback for policy optimization.

3.2 Graph-Based Population Representation

At generation t, the current population contains N candidate solutions evaluated on M optimization objectives. The normalized objective vector of each solution serves as its initial node feature. A population graph is constructed by treating candidate solutions as nodes and Pareto-dominance relationships as directed edges. A two-layer graph convolutional network (GCN) Kipf & Welling

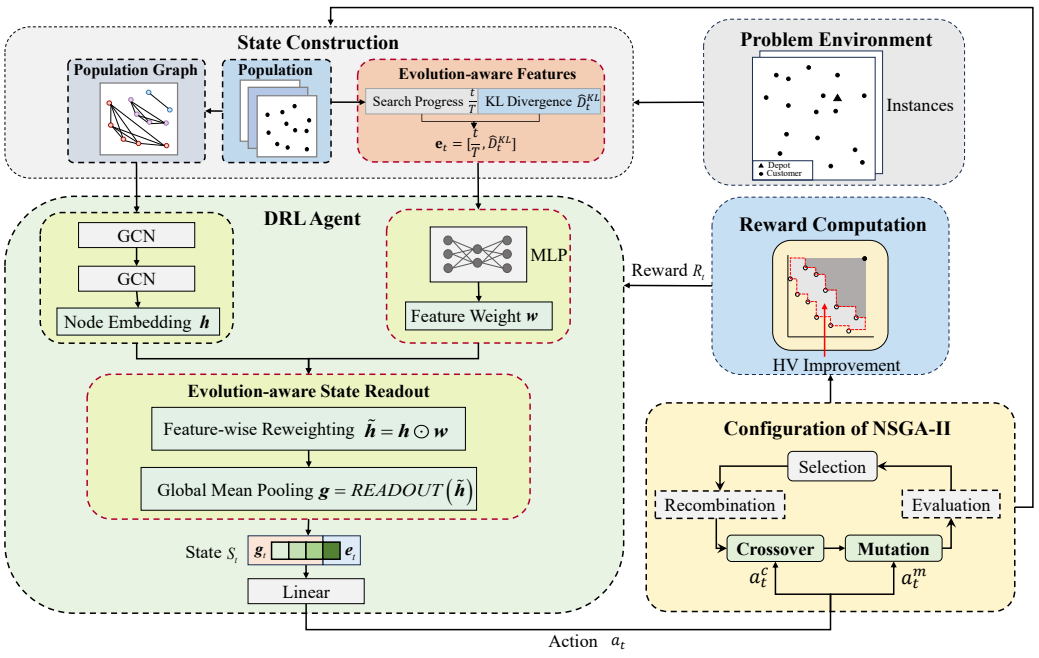


Figure 1: Overview of the proposed evolution-aware dynamic configuration framework.

(2017) is employed to encode the population graph:

$$H_t = \mathrm{GCN}(X_t, E_t), \qquad H_t \in \mathbb{R}^{N \times d} \tag{1}$$

where X_t denotes the normalized node feature matrix, E_t represents the adjacency matrix derived from graph connectivity, and d is the dimension of the latent node embeddings. The i-th row $\boldsymbol{h}_i^t \in \mathbb{R}^d$ represents the structural embedding of the i-th solution and is subsequently used for evolution-guided feature reweighting.

3.3 Evolution-aware Feature Modeling

Although the population graph captures the structure of the current population, it does not explicitly describe inter-generational dynamics. To address this, we introduce the distributional shift between consecutive populations as supplementary evolutionary information. Specifically, the objective vectors at generations $t-1$ and t are modeled as multivariate Gaussian distributions:

$$\mathcal{P}_{t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1}), \qquad \mathcal{P}_t \sim \mathcal{N}(\mu_t, \Sigma_t), \tag{2}$$

where $\boldsymbol{\mu}_{t-1}, \boldsymbol{\mu}_t \in \mathbb{R}^M$ denote the mean objective vectors, and $\boldsymbol{\Sigma}_{t-1}, \boldsymbol{\Sigma}_t \in \mathbb{R}^{M \times M}$ represent the corresponding covariance matrices. To improve numerical stability and avoid singular covariance matrices, a small diagonal regularization term ϵI with $\epsilon = 10^{-6}$ is added to each covariance matrix. The inter-generational distributional change is quantified via the KL divergence:

$$D_t^{\mathrm{KL}} = D_{\mathrm{KL}}\left(\mathcal{P}_{t-1} \,\|\, \mathcal{P}_t\right). \tag{3}$$

For two M-dimensional Gaussian distributions, the closed-form KL divergence is computed as:

$$D_t^{\mathrm{KL}} = \frac{1}{2}\left[\mathrm{tr}\left(\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}_{t-1}\right) - M + \ln\frac{|\boldsymbol{\Sigma}_t|}{|\boldsymbol{\Sigma}_{t-1}|} + \left(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}\right)^\top \boldsymbol{\Sigma}_t^{-1}\left(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}\right)\right]. \tag{4}$$

This metric explicitly measures variations in both population mean position and dispersion within the objective space, where a larger value signifies a more substantial distributional change. We consistently compute $D_{\mathrm{KL}}(\mathcal{P}_{t-1} \| \mathcal{P}_t)$ to preserve temporal ordering. However, this scalar does not

encode the signed direction of population movement. To constrain the unbounded non-negative divergence into a stable numerical range, D_t^{KL} is normalized into $[0, 1)$:

$$\widehat{D}_t^{\mathrm{KL}} = \frac{D_t^{\mathrm{KL}}}{D_t^{\mathrm{KL}} + 1}. \tag{5}$$

Because population changes carry different importance weights across different evolutionary stages, the transformed KL divergence is concatenated with the search progress to yield an evolution-aware feature vector:

$$\boldsymbol{e}_t = \left[\frac{t}{T}, \widehat{D}_t^{\mathrm{KL}} \right], \tag{6}$$

where T represents the total generation budget. The first scalar specifies the temporal search stage, while the second quantifies recent inter-generational progress. Rather than directly determining the algorithm parameters according to $\widehat{D}_t^{\mathrm{KL}}$, this evolution-aware feature is used to guide the representation of the current population.

3.4 Evolution-aware State Representation and Policy Learning

The evolution-aware feature vector is first mapped by a multilayer perceptron (MLP) with a tanh output layer to derive a feature-wise weight vector:

$$\boldsymbol{w}_t = \phi_\psi(\boldsymbol{e}_t), \qquad \boldsymbol{w}_t \in \mathbb{R}^d, \tag{7}$$

where $\phi_\psi(\cdot)$ denotes the weight-generation network parameterized by ψ, and d corresponds to the dimension of the latent node embeddings. The generated vector assigns an adaptive weight to each latent feature dimension and modulates all node embeddings via element-wise multiplication:

$$\widetilde{\boldsymbol{h}}_i^t = \boldsymbol{h}_i^t \odot \boldsymbol{w}_t, \qquad i = 1, \ldots, N, \tag{8}$$

where $\odot$ denotes element-wise multiplication. In this manner, while the spatial structure of the current population remains the core representation target, the contribution of each latent feature dimension is dynamically reweighted according to search progress and recent inter-generational distributional shifts.

The modulated node embeddings are subsequently aggregated via a readout operator to yield the graph-level population representation:

$$\boldsymbol{g}_t = \mathrm{READOUT}\left(\left\{ \widetilde{\boldsymbol{h}}_i^t \right\}_{i=1}^{N} \right). \tag{9}$$

Here, $\mathrm{READOUT}(\cdot)$ denotes the graph-level aggregation operation. In this study, global mean pooling is adopted to aggregate the node embeddings. The policy state $\boldsymbol{s}_t$ is then constructed by concatenating $\boldsymbol{g}_t$ with the evolution-aware feature vector:

$$\boldsymbol{s}_t = [\boldsymbol{g}_t; \boldsymbol{e}_t], \tag{10}$$

where $\boldsymbol{e}_t$ preserves the explicit scalar signals of normalized search progress and consecutive-population KL divergence. Conditioned on $\boldsymbol{s}_t$, the policy network outputs continuous actions specifying the crossover and mutation probabilities:

$$\boldsymbol{a}_t = [a_t^c, a_t^m]^{\mathrm{T}} \sim \pi_\theta\left(\cdot \mid \boldsymbol{s}_t\right), \boldsymbol{a}_t \in [-1, 1]^2. \tag{11}$$

where a_t^{c} and a_t^{m} are the two normalized action components associated with the crossover and mutation probabilities, respectively. The two normalized action components are linearly mapped to the predefined ranges of the crossover and mutation probabilities before being applied to the evolutionary algorithm Coello Coello et al. (2007). The resulting parameter configuration is used for offspring generation at generation t, followed by fitness evaluation and environmental selection to produce the next population P_{t+1}.

Following GS-MODAC Reijnen et al. (2025b), we adopt a normalized HV-improvement reward that assigns a positive value only when the current population exceeds the historical best HV. The detailed formulation is provided in the Appendix C.

Transitions $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ collected from parallel environments are used to estimate returns and GAE, after which PPO jointly optimizes the policy and value networks Schulman et al. (2017). During evaluation, the trained policy selects parameter configurations deterministically without updating network weights.

4 EXPERIMENTS

The proposed method is evaluated on two representative multi-objective combinatorial optimization problems to assess its optimization performance, scalability, and generalization capability. The experiments compare both static and dynamic algorithm configuration methods.

4.1 EXPERIMENTAL SETTING

Benchmark Problems and Instances. We evaluate the proposed method on the Flexible Job Shop Scheduling Problem (FJSP) and the Capacitated Vehicle Routing Problem (CVRP). The FJSP involves both machine assignment and operation sequencing Dauzère-Pérès et al. (2024). Following standard formulations Tamssaouet et al. (2022), we consider five optimization objectives: Makespan, Balanced Workload, Average Flowtime, Total Workload, and Maximum Flowtime. Depending on the number of objectives, these FJSP variants are denoted as Bi-FJSP, Tri-FJSP, and Penta-FJSP, which optimize the first two, first three, and all five objectives, respectively. The CVRP aims to construct optimal routing plans for a fleet of vehicles servicing a set of customers, where each customer exhibits a specific demand and each vehicle operates under a maximum capacity constraint. In this work, the two objectives considered are minimizing the total travel distance and minimizing the maximum route length. This bi-objective setting is denoted as Bi-CVRP.

Following the protocols of GS-MODAC Reijnen et al. (2025b), the FJSP instances are generated using the procedure established by Song et al. (2023), whereas the CVRP instances are generated following the approach of da Costa et al. (2021). For each problem scale and objective setting, 200 instances are generated, with 100 for training and 100 for testing.

Baselines. We compare our method with representative baselines grouped into three categories. In our experiments, all baseline methods are instantiated with NSGA-II as the target evolutionary algorithm to isolate the effects of different parameter configuration strategies.

1. **Fixed-parameter MOEA:** NSGA-II employs a fixed crossover probability of 0.7 and a mutation probability of 0.02 throughout optimization Coello Coello et al. (2007).
2. **Static algorithm configuration method:** irace identifies a well-performing parameter configuration through an offline iterated racing procedure, after which the selected configuration remains fixed during each optimization run López-Ibáñez et al. (2016).
3. **Dynamic algorithm configuration methods:** This category includes MADAC and GS-MODAC. Following the implementation in GS-MODAC, MADAC is adapted to NSGA-II and dynamically selects discrete configurations of crossover and mutation probabilities using multi-agent reinforcement learning Xue et al. (2022); Reijnen et al. (2025b). GS-MODAC represents the current population as a graph in the objective space and employs a graph neural network to encode the search state, based on which a PPO agent continuously adjusts the crossover and mutation probabilities of NSGA-II Reijnen et al. (2025b).

Training. For the FJSP and the CVRP across various problem scales and objective configurations, independent dynamic configuration policies were trained on randomly generated problem instances. The total training budget was set to $1{,}000{,}000$ steps for each FJSP setup and $2{,}500{,}000$ steps for each CVRP setup. NSGA-II served as the target multi-objective evolutionary algorithm, configured with a population size of 50 and a maximum evolutionary budget of 50 generations. The action space comprised two continuous variables representing crossover and mutation probabilities, bounded by problem-specific operational ranges, namely, for FJSP, crossover and mutation probabilities were restricted to $[0.6, 1.0]$ and $[0.0, 0.1]$, respectively, whereas the corresponding ranges for CVRP were set to $[0.5, 0.7]$ and $[0.0, 0.2]$. Policy optimization was performed using PPO, adhering to the core hyperparameter settings of Schulman et al. (2017). Both the actor and critic networks uesd graph neural network-based state encoders with a latent embedding dimension of 64. Data collection leveraged vectorized parallel environments: 5 parallel environments were utilized for FJSP experiments and 10 for CVRP experiments. Total training runtimes ranged from approximately 9 to 27 hours for CVRP and 7 to 72 hours for FJSP.

Testing. For each problem configuration, the trained policy was fixed during testing. We evaluated each method on 100 unseen instances, with 10 independent evolutionary runs per instance using

different random seeds. For each instance, we summarize the resulting HV values using the mean, maximum, and standard deviation, and subsequently report the average of these statistics over all 100 test instances within the same problem configuration. To guarantee consistency, all methods utilize identical predefined reference points for HV calculation. In the resulting performance tables, the highest mean and maximum HV values under each setting are highlighted in bold. An underlined entry indicates that the corresponding method significantly outperforms the baseline method GS-MODAC according to a two-sided Wilcoxon signed-rank test at a significance level of $p < 0.05$.

4.2 Experimental Results

To evaluate the proposed method, the experiments address the following key research questions:

1. **Overall Optimization Performance:** How does the proposed evolution-aware dynamic configuration method compare against static baselines and state-of-the-art dynamic DAC methods across standard benchmark instances?
2. **Cross-size Generalization:** Can a policy trained on instances with a specific size effectively generalize to instances with different sizes without retraining?
3. **Cross-problem Generalization:** How well does the learned configuration policy transfer to unseen structural variants or problems?
4. **Ablation Analysis:** What are the individual contributions of the KL divergence-based inter-generational feature and the evolution-guided feature reweighting mechanism to overall performance?

Overall Optimization Performance. Table 1 shows that the proposed method achieves the highest or joint-highest mean and maximum HV across all 12 problem settings. While competing methods perform comparably on bi-objective FJSP configurations, the proposed method achieves consistent, statistically significant improvements on settings involving larger problem instances or higher-dimensional objective spaces, such as Tri-FJSP-25j5m, Penta-FJSP-25j5m, and Bi-CVRP-500. This observation indicates that static, single-snapshot population representations are often sufficient in simpler search spaces. However, more complex landscapes demand supplementary dynamic cues to distinguish subtle evolutionary states. Compared to default fixed-parameter NSGA-II, these results highlight the necessity of dynamic parameter adaptation. Furthermore, the performance gains over GS-MODAC suggest that incorporating inter-generational distributional information together with evolution-guided feature modulation can improve dynamic configuration. Although the proposed method does not consistently yield the lowest standard deviation, its main strength lies in driving superior overall solution quality and Pareto coverage.

As shown in Figure 2(a), all three methods achieve rapid HV improvements in the early generations, indicating that they quickly identify promising regions of the search space. However, the proposed method maintains more sustained improvement, gradually separating from GS-MODAC and NSGA-II and converging to the highest final HV. In contrast, NSGA-II improves more slowly at later stages, while GS-MODAC stabilizes at a lower HV level. This indicates that the proposed policy continues to provide effective parameter configurations throughout the evolutionary process.

Figure 2(b) presents the final non-dominated solution sets in the three-dimensional objective space. The proposed method covers regions not fully reached by GS-MODAC and NSGA-II and produces a broader distribution of trade-off solutions. Combined with the improved convergence in Figure 2(a), this wider coverage explains its higher final HV. These observations further demonstrate the effectiveness of the evolution-aware representation in improving convergence and solution-set coverage on this representative instance.

Cross-Size Generalization. Table 2 in Appendix D.1 reports the cross-size generalization results on Bi-CVRP. The configuration policies are trained on instances of a specific customer scale and directly evaluated on different scales without retraining or fine-tuning. The suffix of the algorithm name indicates the number of customers. The proposed method significantly outperforms the corresponding GS-MODAC across all cross-size training–testing combinations.

Notably, policies trained on larger customer sizes maintain competitive performance when transferred to smaller-scale instances and vice versa. For instance, the policy trained on Bi-CVRP-500

Table 1: Hypervolume performance of different methods on FJSP test instances with varying objective counts and problem sizes, and on CVRP test instances with varying problem sizes.

	Bi-FJSP-5j5m			Bi-FJSP-10j5m			Bi-FJSP-25j5m		
Method	mean	max	std	mean	max	std	mean	max	std
NSGA-II	1.87×10^4	2.02×10^4	1.21×10^3	3.82×10^4	4.11×10^4	2.29×10^3	9.41×10^4	9.93×10^4	4.84×10^3
irace	$\mathbf{1.92 \times 10^4}$	$\mathbf{2.04 \times 10^4}$	1.06×10^3	3.90×10^4	4.11×10^4	1.95×10^3	9.52×10^4	9.97×10^4	4.03×10^3
MADAC	1.82×10^4	1.95×10^4	7.53×10^2	3.69×10^4	3.98×10^4	4.47×10^3	9.24×10^4	9.72×10^4	3.09×10^3
GS-MODAC	$\mathbf{1.92 \times 10^4}$	$\mathbf{2.04 \times 10^4}$	1.07×10^3	$\mathbf{3.91 \times 10^4}$	4.14×10^4	1.88×10^3	9.51×10^4	9.98×10^4	4.46×10^3
OURS	$\mathbf{1.92 \times 10^4}$	$\mathbf{2.04 \times 10^4}$	1.07×10^3	$\mathbf{3.91 \times 10^4}$	$\mathbf{4.15 \times 10^4}$	1.95×10^3	$\mathbf{9.54 \times 10^4}$	$\mathbf{10.0 \times 10^4}$	4.60×10^3
	Tri-FJSP-5j5m			Tri-FJSP-10j5m			Tri-FJSP-25j5m		
Method	mean	max	std	mean	max	std	mean	max	std
NSGA-II	2.06×10^6	2.22×10^6	1.32×10^5	5.53×10^6	5.95×10^6	3.09×10^5	2.05×10^7	2.18×10^7	1.13×10^6
irace	2.11×10^6	2.26×10^6	1.16×10^5	5.47×10^6	5.82×10^6	2.65×10^5	2.07×10^7	2.20×10^7	1.07×10^6
MADAC	1.99×10^6	2.14×10^6	8.86×10^4	5.39×10^6	5.87×10^6	5.97×10^5	2.09×10^7	2.20×10^7	1.97×10^6
GS-MODAC	2.11×10^6	2.25×10^6	1.17×10^5	5.67×10^6	6.08×10^6	2.98×10^5	2.15×10^7	2.28×10^7	1.05×10^6
OURS	$\mathbf{2.13 \times 10^6}$	$\mathbf{2.27 \times 10^6}$	1.25×10^5	$\mathbf{5.68 \times 10^6}$	$\mathbf{6.09 \times 10^6}$	3.19×10^5	$\mathbf{2.16 \times 10^7}$	$\mathbf{2.29 \times 10^7}$	1.12×10^6
	Penta-FJSP-5j5m			Penta-FJSP-10j5m			Penta-FJSP-25j5m		
Method	mean	max	std	mean	max	std	mean	max	std
NSGA-II	6.01×10^{10}	6.48×10^{10}	3.70×10^9	3.96×10^{11}	4.31×10^{11}	2.42×10^{10}	5.08×10^{12}	5.48×10^{12}	2.75×10^{11}
irace	6.08×10^{10}	6.49×10^{10}	2.98×10^9	4.03×10^{11}	4.38×10^{11}	2.35×10^{10}	5.18×10^{12}	5.63×10^{12}	2.92×10^{11}
MADAC	5.82×10^{10}	6.28×10^{10}	2.72×10^9	3.91×10^{11}	4.29×10^{11}	4.36×10^{10}	5.10×10^{12}	5.74×10^{12}	5.01×10^{11}
GS-MODAC	6.17×10^{10}	6.59×10^{10}	3.13×10^9	$\mathbf{4.14 \times 10^{11}}$	4.48×10^{11}	2.38×10^{10}	5.62×10^{12}	6.07×10^{12}	3.20×10^{11}
OURS	$\mathbf{6.18 \times 10^{10}}$	$\mathbf{6.63 \times 10^{10}}$	3.33×10^9	$\mathbf{4.14 \times 10^{11}}$	$\mathbf{4.50 \times 10^{11}}$	2.52×10^{10}	$\mathbf{5.63 \times 10^{12}}$	$\mathbf{6.12 \times 10^{12}}$	3.46×10^{11}
	Bi-CVRP-100			Bi-CVRP-200			Bi-CVRP-500		
Method	mean	max	std	mean	max	std	mean	max	std
NSGA-II	1.34×10^2	1.47×10^2	7.56	1.56×10^2	1.72×10^2	9.12	2.27×10^2	2.48×10^2	1.28×10^1
irace	1.34×10^2	1.48×10^2	8.02	1.57×10^2	1.72×10^2	9.53	2.27×10^2	2.48×10^2	1.26×10^1
MADAC	1.35×10^2	1.49×10^2	8.01	1.61×10^2	1.76×10^2	9.22	2.33×10^2	2.54×10^2	1.29×10^1
GS-MODAC	1.35×10^2	1.48×10^2	7.95	1.60×10^2	1.76×10^2	9.50	2.35×10^2	2.59×10^2	1.41×10^1
OURS	$\mathbf{1.36 \times 10^2}$	$\mathbf{1.51 \times 10^2}$	8.13	$\mathbf{1.62 \times 10^2}$	$\mathbf{1.79 \times 10^2}$	9.80	$\mathbf{2.42 \times 10^2}$	$\mathbf{2.65 \times 10^2}$	1.36×10^1

achieves identical or competitive mean HV values on Bi-CVRP-100 and Bi-CVRP-200 compared with policies specifically trained on the corresponding target sizes. Conversely, the policy trained on 100-customer instances achieves competitive mean and maximum HV values on Bi-CVRP-200 compared with the policy trained specifically on 200-customer instances. Similarly, the policy trained on Bi-CVRP-200 performs comparably on Bi-CVRP-500 to the policy specifically trained on Bi-CVRP-500. These findings demonstrate the policy's cross-size generalization ability in both directions.

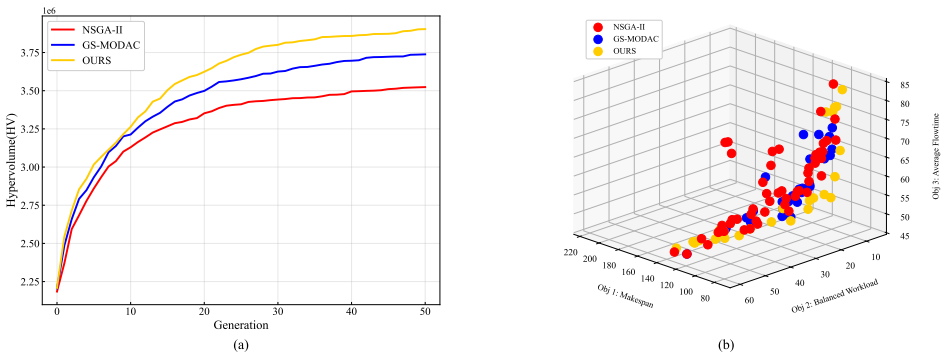


Figure 2: Comparison of OURS, GS-MODAC, and NSGA-II solution methods: (a) Average HV convergence and (b) Pareto front distributions.

Cross-problem Generalization. Table 3 in Appendix D.2 evaluates the zero-shot transfer performance of models trained on standard FJSP-5j5m instances and directly deployed on unseen DAFJS-SDST variants. Compared with standard FJSP, DAFJS-SDST introduces sequence-dependent setup times (SDST), altering the constraint landscape and search structure. This experiment assesses whether the learned configuration policy can adapt to structural problem variations.

The proposed method achieves the highest mean and maximum HV across all three objective settings. Notably, GS-MODAC underperforms the default fixed-parameter NSGA-II on the bi- and tri-objective settings, whereas the proposed method consistently maintains superior performance over both baselines. The results indicate that incorporating inter-generational evolutionary dynamics yields a state representation that is less sensitive to shifts in underlying problem constraints, enabling robust zero-shot transfer. However, the proposed method does not record the lowest standard deviation across these three settings, suggesting that its variance across independent stochastic runs could be further improved. Overall, the findings demonstrate strong cross-problem generalization.

Furthermore, we test the trained policies on 16 unseen instances from CVRPLIB Set X Uchoa et al. (2017), as reported in Table 4 of Appendix D.2. The proposed method outperforms the corresponding GS-MODAC models on most instances and remains competitive across different training and test scales, demonstrating transferability from synthetic training instances to unseen benchmarks and across problem scales.

Ablation Analysis. Table 5 in Appendix D.3 presents the ablation study results on Bi-CVRP instances across varying problem scales. Two ablated variants are considered, i.e., the algorithm with suffix w/o feature-wise reweighting removes the dynamic reweighting of node embedding dimensions, whereas the one with suffix w/o KL guidance retains the reweighting mechanism but excludes the KL divergence signal from the evolution-aware feature. The complete model achieves the highest or joint-highest mean and maximum HV across all evaluated settings, with its performance margin over GS-MODAC expanding as the instance size increases.

Among the ablated variants, removing the feature-wise reweighting mechanism causes the most severe performance degradation, resulting in mean HV values lower than those of GS-MODAC on the 100- and 200-customer instances. This indicates that dynamically modulating the latent feature dimensions of node embeddings serves as a major contributor to overall performance gains. The variant lacking KL guidance remains competitive with the complete model but generally yields slightly lower performance, suggesting that the KL-based inter-generational signal provides complementary information for feature modulation. Although the complete model does not consistently achieve the lowest standard deviation, it delivers the highest or joint-highest mean and maximum HV across the evaluated settings.

5 Conclusion

This paper presents an evolution-aware state representation method for dynamic algorithm configuration in multi-objective combinatorial optimization. By combining the Kullback–Leibler divergence between consecutive population distributions with normalized search progress, the proposed framework constructs a compact evolution-aware feature vector that dynamically guides feature-wise reweighting of latent node embeddings to obtain an evolution-conditioned graph representation. Consequently, the resulting state representation captures both current population topology and inter-generational search dynamics, enabling the reinforcement learning agent to adapt operator parameters more effectively.

Extensive evaluations on multi-objective flexible job shop scheduling and vehicle routing benchmarks demonstrate that the proposed method achieves superior hypervolume performance, faster convergence, and better-distributed solution sets than static and state-of-the-art dynamic configuration baselines. Transfer experiments further show that the learned policy generalizes effectively across different problem scales and unseen instances. Ablation studies confirm that both the KL divergence-based inter-generational feature and evolution-guided feature reweighting contribute substantially to performance gains. Future work will address policy sensitivity under cross-objective transfer and explore adaptive state representations for multi-objective evolutionary algorithms with high-dimensional objectives or constrained environments.

Reproducibility Statement

The main paper and appendices provide the problem formulations, experimental settings, parameter configurations, evaluation protocols, statistical tests, reward formulation, and algorithmic procedure. Additional experimental results, ablation studies, and computational efficiency analyses are provided in the appendices to support reproducibility.

Ethics Statement

This work evaluates optimization algorithms on generated and publicly available benchmark instances and involves no human participants, personal data, or safety-critical deployment. We identify no specific ethical concerns associated with the reported experiments.

AI use statement

This work was completed without the use of generative AI tools. The authors independently prepared and verified the manuscript and all associated research materials.

References

Steven Adriaensen, André Biedenkapp, Gresa Shala, Noor Awad, Theresa Eimer, Marius Lindauer, and Frank Hutter. Automated dynamic algorithm configuration. *Journal of Artificial Intelligence Research*, 75:1633–1699, 2022.

André Biedenkapp, H. Furkan Bozkurt, Theresa Eimer, Frank Hutter, and Marius Lindauer. Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework. In *Proceedings of the European Conference on Artificial Intelligence*, pp. 427–434. IOS Press, 2020.

Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2 edition, 2007.

P da Costa, J Rhuggenaath, Y Zhang, A Akcay, and U Kaymak. Learning 2-opt heuristics for routing problems via deep reinforcement learning. *SN Computer Science*, 2:388, 2021.

Stéphane Dauzère-Pérès, Junwen Ding, Liji Shen, and Karim Tamssaouet. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2):409–432, 2024.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Marcelo Gomes Pereira de Lacerda, Luis Filipe de Araujo Pessoa, Fernando Buarque de Lima Neto, Teresa Bernarda Ludermir, and Herbert Kuchen. A systematic literature review on general parameter control for evolutionary and swarm-based algorithms. *Swarm and Evolutionary Computation*, 60:100777, 2021. ISSN 2210-6502.

Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: an automatic algorithm configuration framework. *Journal of artificial intelligence research*, 36:267–306, 2009.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23 (54):1–9, 2022.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

Robbert Reijnen, Zaharah Bukhsh, Hoong Chuin Lau, Yaoxin Wu, and Yingqian Zhang. Search trajectory network-enhanced multi-objective dynamic algorithm configuration. In *Proceedings of the 28th European Conference on Artificial Intelligence*, volume 413, pp. 4921–4928. IOS Press, 2025a.

Robbert Reijnen, Yaoxin Wu, Zaharah Bukhsh, and Yingqian Zhang. Graph-supported dynamic algorithm configuration for multi-objective combinatorial optimization. In *International Conference on Machine Learning*, volume 267, pp. 51405–51420. PMLR, 13–19 Jul 2025b.

Elias Schede, Jasmin Brandt, Alexander Tornede, Marcel Wever, Viktor Bengs, Eyke Hüllermeier, and Kevin Tierney. A survey of methods for automated algorithm configuration (extended abstract). In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 6964–6968. International Joint Conferences on Artificial Intelligence Organization, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Wen Song, Xinyang Chen, Qiqiang Li, and Zhiguang Cao. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, 2023.

Yanjie Song, Yutong Wu, Yangyang Guo, Ran Yan, Ponnuthurai Nagaratnam Suganthan, Yue Zhang, Witold Pedrycz, Swagatam Das, Rammohan Mallipeddi, Oladayo Solomon Ajani, and Qiang Feng. Reinforcement learning-assisted evolutionary algorithm: A survey and research opportunities. *Swarm and Evolutionary Computation*, 86:101517, 2024.

Karim Tamssaouet, Stéphane Dauzère-Pérès, Sebastian Knopp, Abdoul Bitar, and Claude Yugma. Multiobjective optimization for complex flexible job-shop scheduling problems. *European Journal of Operational Research*, 296(1):87–100, 2022.

Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

Jaime Vergara, Juan Botero, and Luis Fletscher. A comprehensive survey on resource allocation strategies in fog/cloud environments. *Sensors*, 23(9):4413, 2023.

Ke Xue, Jiacheng Xu, Lei Yuan, Miqing Li, Chao Qian, Zongzhang Zhang, and Yang Yu. Multi-agent dynamic algorithm configuration. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20147–20161, 2022.

Sandra Zajac and Sandra Huber. Objectives and methods in multi-objective routing problems: a survey and classification scheme. *European Journal of Operational Research*, 290(1):1–25, 2021.

Wenqiang Zhang, Guanwei Xiao, Mitsuo Gen, Huili Geng, Xiaomeng Wang, Miaolei Deng, and Guohui Zhang. Enhancing multi-objective evolutionary algorithms with machine learning for scheduling problems: recent advances and survey. *Frontiers in Industrial Engineering*, 2: 1337174, 2024.

Yufeng Zhang, Jialu Pan, Li Ken Li, Wanwei Liu, Zhenbang Chen, Xinwang Liu, and Ji Wang. On the properties of Kullback–Leibler divergence between multivariate gaussian distributions. In *Advances in Neural Information Processing Systems*, volume 36, pp. 58152–58165, 2023.

Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011. ISSN 2210-6502.

A TEST PROBLEM CONFIGURATIONS

This appendix provides the formal definitions of the two multi-objective combinatorial optimization problems considered in this study, namely the Flexible Job Shop Scheduling Problem (FJSP) and the Capacitated Vehicle Routing Problem (CVRP). The corresponding objective functions used to construct different experimental settings are also introduced.

A.1 FLEXIBLE JOB SHOP SCHEDULING PROBLEM

The Flexible Job Shop Scheduling Problem considers a set of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and a set of machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Each job J_i consists of an ordered sequence of operations $\mathcal{O}_i = \{O_{i,1}, O_{i,2}, \ldots, O_{i,n_i}\}$, where n_i denotes the number of operations contained in job J_i. Each operation $O_{i,j}$ can be processed by one machine selected from its eligible machine set $\mathcal{M}_{i,j} \subseteq \mathcal{M}$. If operation $O_{i,j}$ is assigned to machine M_k, its processing time is denoted by $p_{i,j,k}$.

The operations belonging to the same job must follow their predefined technological order. Let $S_{i,j}$ and $C_{i,j}$ denote the starting and completion times of operation $O_{i,j}$, respectively. The precedence constraint between two consecutive operations is expressed as

$$S_{i,j+1} \geq C_{i,j}.$$

When $O_{i,j}$ is processed on machine M_k, its completion time satisfies

$$C_{i,j} = S_{i,j} + p_{i,j,k}.$$

Each operation must be assigned to exactly one eligible machine, and each machine can process at most one operation at any time. Moreover, operations cannot be interrupted once processing has started.

To evaluate scheduling quality from different perspectives, five minimization objectives are considered.

- **Makespan.** Makespan represents the completion time of the entire schedule and is defined as
$$C_{\max} = \max_{i=1,\ldots,n} C_{i,n_i}.$$
- **Balanced Workload.** Let $\mathcal{O}_k$ denote the set of operations assigned to machine M_k. The workload of machine M_k is
$$W_k = \sum_{(i,j)\in\mathcal{O}_k} p_{i,j,k}.$$
The workload imbalance among machines is measured by
$$W_{\text{bal}} = \max_{k=1,\ldots,m} W_k - \min_{k=1,\ldots,m} W_k.$$
- **Average flow time.** The flow time of a job is the interval between the start of its first operation and the completion of its final operation. The average flow time is defined as
$$F_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} \left(C_{i,n_i} - S_{i,1}\right).$$
- **Total workload.** The total workload accumulated over all machines is calculated as
$$W_{\text{total}} = \sum_{k=1}^{m} W_k = \sum_{k=1}^{m} \sum_{(i,j)\in\mathcal{O}_k} p_{i,j,k}.$$
- **Maximum flow time.** Maximum flow time measures the longest time spent by any job in the scheduling system:
$$F_{\max} = \max_{i=1,\ldots,n} \left(C_{i,n_i} - S_{i,1}\right).$$

Different subsets of these objectives are selected to construct the bi-objective, tri-objective, and five-objective FJSP configurations used in the experiments. These configurations evaluate whether the dynamic configuration policy remains effective as the number of conflicting optimization objectives increases.

A.2 CAPACITATED VEHICLE ROUTING PROBLEM.

The Capacitated Vehicle Routing Problem (CVRP) considers the delivery of goods from a central depot to a set of geographically distributed customers using a fleet of capacity-constrained vehicles. Let $\mathcal{N} = \{1, \ldots, n\}$ denote the set of customers, and let the depot be indexed by 0. The set of all locations, including the depot and customers, is denoted by $\mathcal{N}_0 = \{0\} \cup \mathcal{N}$. Each customer $i \in \mathcal{N}$ has a non-negative demand q_i. Let $\mathcal{K} = \{1, \ldots, K\}$ denote the set of identical vehicles, each with a maximum carrying capacity Q.

A feasible solution consists of a set of vehicle routes. Each route starts from the depot, visits a subset of customers, and finally returns to the depot. Every customer must be visited exactly once by one vehicle, and the total demand assigned to any vehicle cannot exceed its capacity. Let y_{ik} be a binary variable indicating whether customer $i \in \mathcal{N}$ is served by vehicle $k \in \mathcal{K}$. The capacity requirement is expressed as

$$\sum_{i \in \mathcal{N}} q_i y_{ik} \leq Q, \qquad \forall k \in \mathcal{K}.$$

Let d_{ij} denote the travel distance from location i to location j, where $i, j \in \mathcal{N}_0$ and $i \neq j$. The distance may be derived from the geographical coordinates of the locations or obtained from an actual transportation network. Let x_{ijk} be a binary decision variable defined as

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ travels directly from location } i \text{ to location } j, \\ 0, & \text{otherwise,} \end{cases}$$

where $i, j \in \mathcal{N}_0$, $i \neq j$, and $k \in \mathcal{K}$.

The bi-objective CVRP considered in this study simultaneously minimizes the total travel distance of all vehicles and the length of the longest individual route.

- **Total travel distance.** The total distance traveled by all vehicles is defined as

$$D_{\text{total}} = \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}_0} \sum_{\substack{j \in \mathcal{N}_0 \\ j \neq i}} d_{ij} x_{ijk}.$$

- **Longest route.** The distance traveled by vehicle $k \in \mathcal{K}$ is defined as

$$D_k = \sum_{i \in \mathcal{N}_0} \sum_{\substack{j \in \mathcal{N}_0 \\ j \neq i}} d_{ij} x_{ijk}.$$

  The longest-route objective is consequently expressed as

$$D_{\max} = \max_{k \in \mathcal{K}} D_k.$$

Minimizing D_{total} improves overall transportation efficiency, whereas minimizing $D_{\max}$ avoids excessively long individual routes and promotes a more balanced distribution of travel workloads among vehicles. Since reducing the total travel distance may conflict with balancing individual route lengths, these two objectives result in a set of trade-off solutions.

B PERFORMANCE METRIC

Evaluating the performance of a multi-objective optimization algorithm requires considering both the convergence and diversity of the obtained non-dominated solutions. Convergence reflects how closely the approximate Pareto front approaches the true Pareto front, whereas diversity measures how widely and uniformly the solutions cover different regions of the objective space. Therefore, assessing only a single objective value or a representative solution is insufficient to characterize the overall quality of an approximate Pareto front.

In this study, the hypervolume (HV) indicator is adopted as the primary performance metric. HV is widely used in multi-objective optimization because it simultaneously captures convergence and

diversity within a single scalar value. Moreover, unlike several distance-based indicators, its computation does not require prior knowledge of the true Pareto front, which is generally unavailable for the multi-objective combinatorial optimization problems considered in this work. Therefore, HV provides a suitable and comprehensive measure for comparing the overall optimization performance of different methods.

Given a non-dominated objective set $\mathcal{Y}$ and a reference point $\boldsymbol{r} \in \mathbb{R}^M$, the HV is defined as

$$HV_{\boldsymbol{r}}(\mathcal{Y}) = \mu\left(\bigcup_{\boldsymbol{y}\in\mathcal{Y}} [\boldsymbol{y}, \boldsymbol{r}]\right),$$

where $\mu(\cdot)$ denotes the Lebesgue measure, i.e., the M-dimensional volume in the objective space. For a minimization problem, $\boldsymbol{r}$ is selected as a fixed point that is component-wise worse than all evaluated objective vectors. The expression$[\boldsymbol{y}, \boldsymbol{r}] = [y_1, r_1] \times \cdots \times [y_M, r_M]$ denotes the M-dimensional hyperrectangle bounded by $\boldsymbol{y}$ and $\boldsymbol{r}$.

A larger HV value indicates that the obtained non-dominated solutions are, in general, closer to the Pareto-optimal region and cover a broader range of trade-offs. To ensure a fair comparison, the same reference point is used for all methods under each experimental setting.

C Dynamic Configuration and Policy Learning.

Algorithm 1 summarizes the interaction between the reinforcement learning agent and NSGA-II, together with the policy-learning procedure. At each generation, the evolution-aware state is constructed from the current population and used by the policy to determine the crossover and mutation probabilities. The configured evolutionary operators are then applied to generate the next population, and an HV-based reward is calculated to evaluate the effectiveness of the current parameter configuration. Transitions collected from the parallel environments are subsequently used to estimate the discounted returns and generalized advantage estimates (GAE). Based on these estimates, the actor, critic, GCN encoder, and feature-weight generator are jointly updated using proximal policy optimization (PPO). For each training instance I, an approximate ideal hypervolume $HV_{\text{ideal}}(I)$ is obtained offline using NSGA-II before policy training and is subsequently used to normalize the HV-based reward.

D Detailed Experimental Results

This appendix shows the complete results in Table 2, Table 3, Table 4 and Table 5. The main text presents the principal experimental findings, while the following discussion provides additional observations on convergence behavior and Cross-problem Generalization.

D.1 Cross-Size Generalization

The complete experimental results for cross-size generalization are reported in Table 2.

Table 2: Cross-size generalization on Bi-CVRP. Models trained on one customer size are evaluated on unseen instances with other size customers.

	Bi-CVRP-100			Bi-CVRP-200			Bi-CVRP-500		
Method	mean	max	std	mean	max	std	mean	max	std
NSGA-II	1.34×10^2	1.47×10^2	7.84	1.56×10^2	1.72×10^2	9.05	2.27×10^2	2.48×10^2	1.27×10^1
GS-MODAC - 100	1.35×10^2	1.48×10^2	7.92	1.59×10^2	1.74×10^2	9.20	2.32×10^2	2.55×10^2	1.31×10^1
OURS - 100	$\mathbf{1.36 \times 10^2}$	$\mathbf{1.51 \times 10^2}$	8.18	1.61×10^2	1.77×10^2	9.26	2.37×10^2	2.60×10^2	1.34×10^1
GS-MODAC - 200	1.35×10^2	1.49×10^2	8.40	1.60×10^2	1.76×10^2	9.50	2.33×10^2	2.56×10^2	1.36×10^1
OURS - 200	$\mathbf{1.36 \times 10^2}$	1.50×10^2	8.18	$\mathbf{1.62 \times 10^2}$	$\mathbf{1.79 \times 10^2}$	9.26	2.41×10^2	2.62×10^2	1.37×10^1
GS-MODAC - 500	1.33×10^2	1.47×10^2	8.52	1.60×10^2	1.75×10^2	9.33	2.36×10^2	2.59×10^2	1.41×10^1
OURS - 500	$\mathbf{1.36 \times 10^2}$	1.49×10^2	8.13	$\mathbf{1.62 \times 10^2}$	$\mathbf{1.79 \times 10^2}$	9.80	$\mathbf{2.42 \times 10^2}$	$\mathbf{2.65 \times 10^2}$	1.36×10^1

D.2 CROSS-PROBLEM GENERALIZATION

The complete cross-problem generalization results are reported in Tables 3 and 4. Since the results in Table 3 have been discussed in detail in the main text, the following analysis focuses primarily on the additional transfer results presented in Table 4.

Table 4 reports the mean HV on 16 unseen CVRPLIB Set X instances Uchoa et al. (2017). Compared with GS-MODAC trained at the same scale, our method obtains higher mean HV on 15, 14, and 15 instances for the 100-, 200-, and 500-customer training settings, respectively, with one tie for the 500-customer model. Compared with fixed-parameter NSGA-II, the corresponding numbers are 16, 15, and 16. These results show that the learned policies retain strong numerical performance when transferred from synthetic training instances to unseen benchmark instances.

Moreover, one of the proposed models achieves the highest or joint-highest mean HV on every test instance. The 100-, 200-, and 500-customer models attain or share the best result on 5, 6, and 6 instances, respectively. Among the three training scales, the 500-customer policy shows the most consistent advantage over its corresponding GS-MODAC baseline, achieving higher mean HV on 15 instances and tying on the remaining one. Although models trained on larger scales frequently perform well on larger test instances, the correspondence is not strict, and the best-performing training scale varies across instances rather than following a monotonic relationship with instance size. At the same time, the few cases where the transferred policy does not outperform its matched GS-MODAC baseline indicate that the advantage is broad but not uniform across all training-test scale combinations. Overall, these results suggest that the learned policies are not restricted to their training scale and exhibit cross-scale transferability.

Table 3: Comparison of the generalization performance of trained models on unseen instances of more complex problem variants.

	Bi-DAFJS-SDST			Tri-DAFJS-SDST			Penta-DAFJS-SDST		
Method	mean	max	std	mean	max	std	mean	max	std
NSGA-II	1.35×10^{6}	1.45×10^{6}	7.69×10^{4}	3.31×10^{8}	3.56×10^{8}	1.76×10^{7}	7.55×10^{14}	8.33×10^{14}	4.92×10^{13}
GS-MODAC-FJSP-5j5m	1.32×10^{6}	1.41×10^{6}	6.31×10^{4}	3.10×10^{8}	3.38×10^{8}	1.85×10^{7}	8.49×10^{14}	9.21×10^{14}	5.17×10^{13}
OURS-FJSP-5j5m	$\mathbf{1.39 \times 10^{6}}$	$\mathbf{1.48 \times 10^{6}}$	7.10×10^{4}	$\mathbf{3.53 \times 10^{8}}$	$\mathbf{3.82 \times 10^{8}}$	2.08×10^{7}	$\mathbf{8.72 \times 10^{14}}$	$\mathbf{9.68 \times 10^{14}}$	5.64×10^{13}

Table 4: Cross-problem generalization on unseen CVRPLIB Set X instances.

		Trained on $n = 100$		Trained on $n = 200$		Trained on $n = 500$	
Set-X-instance	NSGA-II	GS-MODAC	OURS	GS-MODAC	OURS	GS-MODAC	OURS
set-X-n101	1.75×10^{7}	1.76×10^{7}	$\mathbf{1.78 \times 10^{7}}$	1.75×10^{7}	1.73×10^{7}	1.68×10^{7}	1.76×10^{7}
set-X-n110	4.84×10^{6}	4.81×10^{6}	$\mathbf{5.56 \times 10^{6}}$	5.03×10^{6}	5.48×10^{6}	4.79×10^{6}	4.99×10^{6}
set-X-n125	1.30×10^{7}	1.27×10^{7}	1.35×10^{7}	1.32×10^{7}	$\mathbf{1.39 \times 10^{7}}$	1.36×10^{7}	1.36×10^{7}
set-X-n143	1.74×10^{7}	1.74×10^{7}	$\mathbf{1.90 \times 10^{7}}$	1.78×10^{7}	1.84×10^{7}	1.80×10^{7}	1.90×10^{7}
set-X-n181	3.16×10^{6}	3.40×10^{6}	$\mathbf{3.80 \times 10^{6}}$	3.46×10^{6}	3.72×10^{6}	2.93×10^{6}	3.32×10^{6}
set-X-n195	1.99×10^{7}	2.09×10^{7}	2.14×10^{7}	2.11×10^{7}	$\mathbf{2.24 \times 10^{7}}$	2.06×10^{7}	2.10×10^{7}
set-X-n200	1.60×10^{7}	1.67×10^{7}	1.69×10^{7}	1.56×10^{7}	$\mathbf{1.73 \times 10^{7}}$	1.60×10^{7}	1.62×10^{7}
set-X-n223	3.25×10^{7}	3.56×10^{7}	3.58×10^{7}	3.31×10^{7}	3.52×10^{7}	3.52×10^{7}	$\mathbf{3.77 \times 10^{7}}$
set-X-n491	8.55×10^{7}	9.28×10^{7}	9.32×10^{7}	8.94×10^{7}	$\mathbf{9.52 \times 10^{7}}$	9.26×10^{7}	9.51×10^{7}
set-X-n513	4.47×10^{7}	4.76×10^{7}	4.70×10^{7}	4.75×10^{7}	4.89×10^{7}	4.81×10^{7}	$\mathbf{4.99 \times 10^{7}}$
set-X-n524	8.80×10^{7}	9.18×10^{7}	9.24×10^{7}	9.39×10^{7}	$\mathbf{9.45 \times 10^{7}}$	9.10×10^{7}	9.30×10^{7}
set-X-n548	3.45×10^{7}	3.44×10^{7}	$\mathbf{3.75 \times 10^{7}}$	3.50×10^{7}	3.72×10^{7}	3.45×10^{7}	3.59×10^{7}
set-X-n561	4.69×10^{7}	4.91×10^{7}	4.99×10^{7}	5.08×10^{7}	4.83×10^{7}	4.90×10^{7}	$\mathbf{5.11 \times 10^{7}}$
set-X-n573	5.92×10^{7}	6.08×10^{7}	6.28×10^{7}	6.15×10^{7}	6.32×10^{7}	6.23×10^{7}	$\mathbf{6.43 \times 10^{7}}$
set-X-n627	7.58×10^{7}	7.58×10^{7}	7.96×10^{7}	7.58×10^{7}	7.81×10^{7}	7.68×10^{7}	$\mathbf{8.30 \times 10^{7}}$
set-X-n641	4.24×10^{7}	4.26×10^{7}	4.57×10^{7}	4.45×10^{7}	$\mathbf{4.69 \times 10^{7}}$	4.31×10^{7}	4.51×10^{7}

D.3 Ablation Analysis

The complete experimental results for ablation analysis are reported in Table 5.

Table 5: Ablation study on Bi-CVRP

	Bi-CVRP-100			Bi-CVRP-200			Bi-CVRP-500		
Method	mean	max	std	mean	max	std	mean	max	std
GS-MODAC	1.35×10^2	1.48×10^2	7.95	1.60×10^2	1.76×10^2	9.50	2.35×10^2	2.59×10^2	1.41×10^1
OURS(w/o feature-wise reweight)	1.34×10^2	1.48×10^2	8.13	1.59×10^2	1.74×10^2	9.21	2.36×10^2	2.59×10^2	1.36×10^1
OURS(w/o KL guidance)	1.36×10^2	1.49×10^2	8.05	1.61×10^2	1.77×10^2	9.14	2.41×10^2	2.65×10^2	1.37×10^1
OURS(FULL)	$\mathbf{1.36 \times 10^2}$	$\underline{\mathbf{1.51 \times 10^2}}$	8.13	$\underline{\mathbf{1.62 \times 10^2}}$	$\underline{\mathbf{1.79 \times 10^2}}$	9.80	$\underline{\mathbf{2.42 \times 10^2}}$	$\mathbf{2.65 \times 10^2}$	1.36×10^1

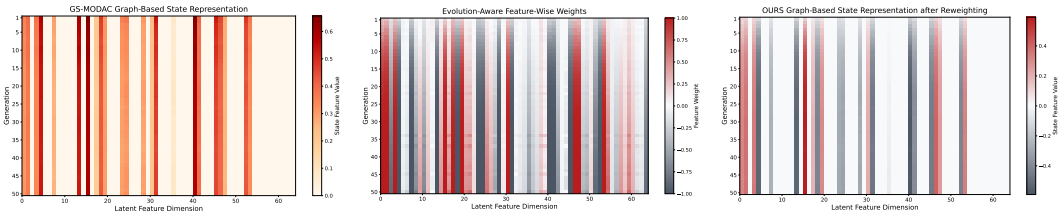
E Visualization of Evolution-Guided Feature Reweighting

Figure 3 visualizes how the proposed feature-wise reweighting mechanism modifies the graph-based state representation across generations. Figure 3(a) shows the representation obtained by GS-MODAC. Specifically, the current population graph is encoded by a two-layer GCN, and the resulting node embeddings are directly aggregated through global mean pooling without the evolution-guided feature-wise reweighting introduced in our method. This representation also serves as the base graph representation in the proposed method and therefore provides a direct reference for examining the effect of the additional feature-wise reweighting mechanism.

As shown in Figure 3(a), the graph-based state representation exhibits a relatively stable vertical-band pattern across generations. Only a subset of latent dimensions remains strongly activated, whereas many dimensions stay close to zero throughout the evolutionary process. This indicates that the representation used by GS-MODAC captures the structural characteristics of the current population, but the relative contributions of different latent dimensions vary only mildly across generations.

Figure 3(b) presents the feature-wise weights generated from the evolution-aware feature, which consists of the normalized search progress and the KL divergence between consecutive populations. The weights exhibit clear variations across both latent dimensions and generations. This indicates that the proposed mechanism does not apply the same adjustment to all embedding dimensions. Instead, it dynamically changes the contribution of each dimension according to the current search stage and recent inter-generational changes. Positive weights preserve the sign of the corresponding activations, values close to zero weaken the corresponding dimensions, and negative weights change the signs of the activations.

Figure 3(c) shows the graph-based state representation obtained by OURS after applying the generated weights to the node embeddings and subsequently performing global mean pooling. Compared with the GS-MODAC-style representation in Figure 3(a), the activation pattern is visibly modified: some dimensions are strengthened, while others are weakened or change sign. The resulting representation also exhibits more noticeable variations across generations. These observations illustrate that the proposed mechanism extends the GS-MODAC state-representation pipeline by incorporating inter-generational evolutionary information into the latent embedding dimensions before graph-level aggregation. Consequently, the graph-based component of the constructed state becomes more responsive to changes occurring during the evolutionary process.



(a) (b) (c)

Figure 3: Visualization of the evolution-guided feature reweighting process: (a) the graph-based state representation obtained by GS-MODAC; (b) the feature-wise weights generated from the evolution-aware feature; and (c) the graph-based state representation obtained by OURS after feature-wise reweighting.

F Computational Efficiency Analysis

All training and evaluation runs were conducted on the same computing node equipped with an NVIDIA GeForce RTX 4090 D GPU. Table 6 compares training time of GS-MODAC and the proposed method, with both methods trained for the same number of environment steps.The proposed method requires 9.7, 12.3, and 27.8 hours on Bi-CVRP-100, Bi-CVRP-200, and Bi-CVRP-500, respectively, corresponding to increases of 14.12%, 20.59%, and 32.38% over GS-MODAC. The increasing gap indicates that the additional training cost becomes more pronounced as the problem size grows. This is reasonable because larger instances involve more node embeddings during evolution-guided feature reweighting, together with the additional computation of inter-generational KL divergence and feature-wise weight generation. Nevertheless, the reported time reflects the overall training process, including evolutionary simulation, trajectory collection, data transfer, and policy optimization, rather than the isolated cost of the proposed modules.

Table 6: Training Time Comparison

	Bi-CVRP-100	Bi-CVRP-200	Bi-CVRP-500
GS-MODAC	8.5 h	10.2 h	21.0 h
OURS	9.7 h	12.3 h	27.8 h
Relative change	$+14.12\%$	$+20.59\%$	$+32.38\%$

Table 7 further reports the total test runtime required to complete a full evolutionary run. Compared with GS-MODAC, the relative runtime changes of the proposed method are only 1.05%, -5.86%, and 2.31% on Bi-CVRP-100, Bi-CVRP-200, and Bi-CVRP-500, respectively. All variations remain within 6%, and the corresponding standard deviations are also comparable.

These results indicate that although the proposed method incurs additional offline training cost, the computation of the inter-generational KL divergence and evolution-guided feature reweighting introduces only limited overhead during testing.

Table 7: Total Test Runtime per Evolutionary Run

	Bi-CVRP-100		Bi-CVRP-200		Bi-CVRP-500	
Method	mean	std	mean	std	mean	std
GS-MODAC	$1.91s$	0.29	$2.90s$	0.23	$6.05s$	0.23
OURS	$1.93s$	0.25	$2.73s$	0.21	$6.19s$	0.27
Relative change (%)	$+1.05$		-5.86		$+2.31$	

Algorithm 1 Framework of the Evolution-Aware DAC Method

Input: Training instance set $\mathcal{I}$; population size N; number of generations T per episode; predefined crossover range $[p^c_{\min}, p^c_{\max}]$; predefined mutation range $[p^m_{\min}, p^m_{\max}]$; estimated ideal hypervolume $HV_{\text{ideal}}(I)$ for each $I \in \mathcal{I}$; policy-training termination criterion.
Output: Trained dynamic configuration policy π_θ.

Initialize parallel evolutionary environments, actor π_θ, critic V_ω, GCN encoder, feature-weight generator ϕ_ψ, and trajectory buffer $\mathcal{D} \leftarrow \emptyset$
while the policy-training termination criterion is not met **do**
  Clear the trajectory buffer $\mathcal{D} \leftarrow \emptyset$
  for each episode collected from the parallel environments **do**
    Sample a training instance I from $\mathcal{I}$ and obtain its corresponding $HV_{\text{ideal}}(I)$
    Initialize and evaluate a population P_0 of size N
    Compute $HV_{\text{initial}} \leftarrow HV(P_0)$ and set $HV_{\text{best}} \leftarrow HV_{\text{initial}}$
    Initialize the normalized inter-generational divergence $\widehat{D}_0^{KL} \leftarrow 0$
    for $t = 0$ to $T-1$ **do**
      Construct the Pareto-dominance graph $(\mathbf{X}_t, \mathbf{E}_t)$ from P_t and obtain node embeddings $\mathbf{H}_t \leftarrow \text{GCN}(\mathbf{X}_t, \mathbf{E}_t)$ (Eq. 1)
      Construct the evolution-aware feature $\mathbf{e}_t \leftarrow [t/T, \widehat{D}_t^{KL}]$ (Eq. 6)
      Generate feature-wise weights $\mathbf{w}_t \leftarrow \phi_\psi(\mathbf{e}_t)$ and compute $\mathbf{g}_t \leftarrow \text{READOUT}(\mathbf{H}_t \odot \mathbf{w}_t)$ (Eqs. 7–9)
      Construct the policy state $\mathbf{s}_t \leftarrow [\mathbf{g}_t; \mathbf{e}_t]$ (Eq. 10)
      Sample a normalized action $\mathbf{a}_t = [a_t^c, a_t^m] \sim \pi_\theta(\cdot \mid \mathbf{s}_t)$, where $\mathbf{a}_t \in [-1, 1]^2$; (Eq. 11)
      Map the action to the parameter values $p_t^u \leftarrow p^u_{\min} + \dfrac{a_t^u + 1}{2}(p^u_{\max} - p^u_{\min}),\ u \in \{c, m\}$
      Generate and evaluate offspring using crossover probability p_t^c and mutation probability p_t^m
      Perform NSGA-II environmental selection to obtain the next population P_{t+1}
      Compute the inter-generational KL divergence $D_{t+1}^{KL} \leftarrow D_{\text{KL}}(P_t \| P_{t+1})$ (Eqs. 3–4)
      Normalize the KL divergence $\widehat{D}_{t+1}^{KL} \leftarrow \dfrac{D_{t+1}^{KL}}{D_{t+1}^{KL} + 1}$ (Eq. 5)
      Compute the current hypervolume $HV_{\text{current}} \leftarrow HV(P_{t+1})$
      Compute the normalized current HV improvement $\Delta_t^{\text{current}} \leftarrow \dfrac{HV_{\text{current}} - HV_{\text{initial}}}{HV_{\text{ideal}}(I) - HV_{\text{initial}}} \times 100$
      Compute the normalized best-so-far HV improvement $\Delta_t^{\text{best}} \leftarrow \dfrac{HV_{\text{best}} - HV_{\text{initial}}}{HV_{\text{ideal}}(I) - HV_{\text{initial}}} \times 100$
      if $HV_{\text{current}} > HV_{\text{best}}$ **then**
        Set the HV-based reward $r_t \leftarrow (\Delta_t^{\text{current}})^2 - (\Delta_t^{\text{best}})^2$
      else
        Set $r_t \leftarrow 0$
      end if
      Update the best-so-far hypervolume $HV_{\text{best}} \leftarrow \max\{HV_{\text{best}}, HV_{\text{current}}\}$
      Construct the next state $\mathbf{s}_{t+1}$ from P_{t+1}, $(t+1)/T$, and $\widehat{D}_{t+1}^{KL}$ using the same state-construction procedure
      Store the transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ in $\mathcal{D}$
    end for
  end for
  Estimate the discounted returns and GAE from the trajectories stored in $\mathcal{D}$
  Update the actor π_θ, critic V_ω, GCN encoder, and feature-weight generator ϕ_ψ using PPO
end while
return π_θ