MOSAIC: MODULAR SEARCH ARCHITECTURE EVOLUTION FOR LLM-DRIVEN HEURISTIC DESIGN IN MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-driven automated heuristic design (AHD) has attracted increasing attention for its potential to reduce labor-intensive manual design. However, existing LLM-driven multi-operator AHD methods typically optimize search behaviors within predefined operator organizations, which constrains the exploration of potential synergies among different search behaviors. To address this limitation, we propose MOSAIC, a modular search architecture evolution framework for LLM-driven AHD. MOSAIC represents each heuristic as an ordered, variable-length architecture composed of multiple search modules, where each module implements a search behavior and the architecture determines how these behaviors are organized. Based on this representation, MOSAIC jointly evolves modules and architectures through module rewriting and architecture variation. The former rewrites modules to explore and refine search behaviors, while the latter evolves module composition, execution order, and architecture length, enabling broader exploration of potential synergies among search behaviors. Experiments on multiple multi-objective combinatorial optimization problems show that MOSAIC outperforms advanced LLM-driven AHD methods and exhibits strong cross-scale generalization.

1 INTRODUCTION

Combinatorial optimization problems (COPs) aim to find optimal solutions that satisfy given constraints within discrete feasible solution spaces (Juan et al., 2015). Many COPs are NP-hard, making it difficult for exact algorithms to obtain optimal solutions within a reasonable time. Heuristic methods can find high-quality solutions under limited computational budgets and are widely used to solve COPs (Baghel et al., 2012). However, designing effective heuristics for specific COPs typically requires expert knowledge and costly trial and error (Liu et al., 2026c).

In recent years, large language model (LLM)-driven automated heuristic design (AHD) has emerged as a promising paradigm for heuristic design (Liu et al., 2026b; Chen et al., 2026; Huang et al., 2026; Corrêa et al., 2025). A common strategy in LLM-driven AHD is to use the LLM as an algorithm designer embedded within an iterative search framework, where search feedback guides heuristic evolution (Liu et al., 2026c). Such search frameworks include evolutionary computation (EC) (Liu et al., 2024; Ye et al., 2024; Dat et al., 2025; Yao et al., 2025; Xu et al., 2026), monte carlo tree search (MCTS) (Zheng et al., 2025; Wang et al., 2025a;b; Qiu et al., 2026), neighborhood search (NS) (Xie et al., 2025), and large neighborhood search (LNS) (Zhao et al., 2026).

Most existing LLM-driven AHD methods have focused on the design of individual search operators (Liu et al., 2026c). However, complex multi-objective combinatorial optimization problems (MCOPs) require the search process to handle multiple conflicting objectives while balancing exploration and exploitation (Qiu et al., 2026; Guan et al., 2024). A single operator typically exhibits a specific search preference, making it difficult to accommodate the diverse search requirements of MCOPs (Qiu et al., 2026; Tian et al., 2022).

Recent LLM-driven multi-operator AHD methods address this challenge by jointly designing operators with complementary search behaviors. Their design spaces, however, primarily concern operator behaviors, while the organization of these behaviors is prescribed by the underlying search

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.

structure. E2OC (Qiu et al., 2026) predefines a set of operator slots and their execution order, and searches for promising design strategy combinations across these slots. G-LNS (Zhao et al., 2026) adopts a fixed Destroy–Repair structure and separately generates and evolves operators for the two predefined roles. Although these methods expand the space of LLM-generated search behaviors, their potential synergies remain conditioned on predefined organizational structures. This observation suggests that the organization of search behaviors should itself be treated as a design variable. Jointly evolving behaviors and their organization therefore enables broader exploration of potential synergies among them.

To this end, we propose MOSAIC, a modular search architecture evolution framework for LLM-driven AHD. MOSAIC represents a heuristic as an ordered, variable-length architecture of search modules, treating module behaviors and their organization as distinct but jointly optimized design dimensions. Based on this representation, MOSAIC jointly evolves modules and architectures through two complementary mechanisms. LLM-driven module rewriting explores the behavioral space of individual modules, while architecture variation explores their composition, execution order, and architecture length. Candidate architectures are evaluated as complete heuristics, so that module behaviors and their organization are jointly assessed based on the resulting search performance.

The main contributions are summarized as follows:

- We propose MOSAIC, a modular search architecture evolution framework that represents heuristics as variable-length architectures of search modules, introducing operator organization as an explicit design dimension in LLM-driven multi-operator AHD.
- We develop a joint module–architecture evolution mechanism that combines LLM-driven module rewriting with architecture variation to jointly optimize search behaviors and their organization.
- We comprehensively evaluate MOSAIC on multiple MCOPs. Results show that MOSAIC outperforms advanced LLM-driven single-operator and multi-operator AHD methods and exhibits strong cross-scale generalization.

2 METHODOLOGY

2.1 MOSAIC OVERVIEW

We propose MOSAIC, a modular search architecture evolution framework for LLM-driven AHD in MCOPs. MOSAIC represents each heuristic as a modular search architecture and jointly evolves its modules and architecture. A search architecture is defined as an ordered, variable-length composition of search modules. Each module implements a search behavior, while the architecture determines the composition and execution order of these behaviors.

The overall framework of MOSAIC is illustrated in Figure 1. MOSAIC first constructs an initial module pool and architecture population. It then iteratively generates candidate architectures through LLM-driven module rewriting and architecture variation. Module rewriting explores and refines search behaviors, while architecture variation evolves module composition, execution order, and architecture length. Candidate architectures are evaluated as complete heuristics on the training instances and undergo multi-objective survival selection. The surviving architectures and their modules are retained for subsequent evolution.

2.2 MODULE POOL AND ARCHITECTURE INITIALIZATION

The initialization of MOSAIC consists of two stages: module pool construction and architecture population formation. The former constructs a set of reusable search modules with diverse search behaviors, while the latter composes these modules into candidate architectures.

Module Pool Construction. MOSAIC encodes the task description and module interface specification into an initialization prompt. The module interface specification constrains only the input and output formats of the modules, without prescribing their search semantics or specific implementations. This allows diverse search behaviors to be implemented as reusable modules under a unified interface and composed within different architectures.

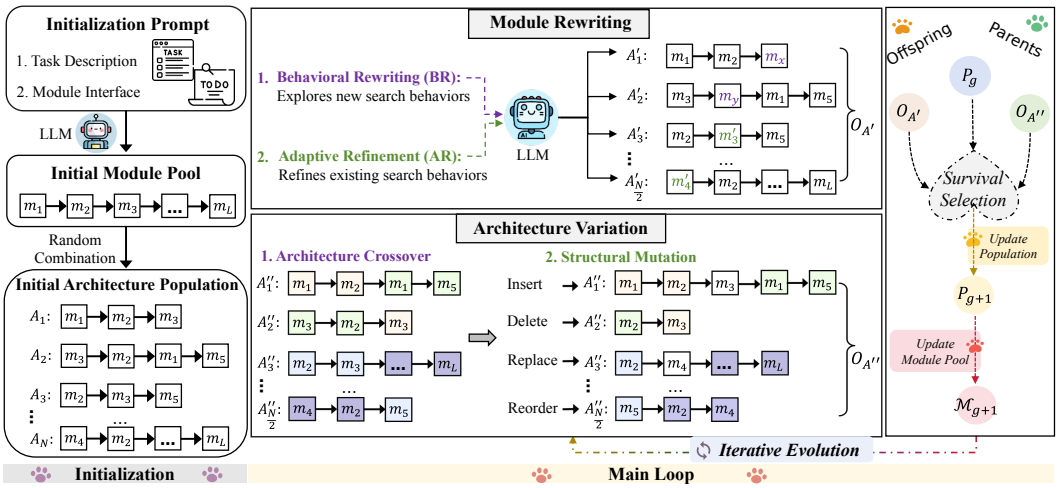


Figure 1: Overview of the MOSAIC framework. An initial module pool is generated and used to construct the initial architecture population. During evolution, module rewriting explores and refines search behaviors, while architecture variation evolves module composition, execution order, and architecture length. The resulting architectures undergo survival selection to update the architecture population and module pool.

Given the initialization prompt, MOSAIC invokes the LLM to generate candidate modules, which undergo validity and duplication checks. Valid modules are integrated into the underlying optimization framework and evaluated on the training instances. The top-performing K modules are retained to form the initial module pool $\mathcal{M}_0 = \{m_1, \ldots, m_K\}$.

Architecture Population Formation. Given the initial module pool $\mathcal{M}_0$, MOSAIC represents each candidate heuristic as an ordered architecture with variable length,

$$\mathcal{A} = (m_{i_1}, m_{i_2}, \ldots, m_{i_L}), \qquad 1 \le L \le L_{\max}, \tag{1}$$

where $i_j \in \{1, \ldots, |\mathcal{M}_0|\}$ denotes the module index at the jth position, L is the architecture length, and $L_{\max}$ is the maximum length.

Based on this representation, MOSAIC searches over architectures with different module compositions, execution orders, and architecture lengths. More generally, given a module pool $\mathcal{M}$, the architecture space is defined as

$$\mathcal{H}(\mathcal{M}) = \bigcup_{L=1}^{L_{\max}} \mathcal{M}^L, \tag{2}$$

where $\mathcal{M}^L$ denotes the set of all ordered module sequences of length L.

Candidate architectures are sampled from $\mathcal{H}(\mathcal{M}_0)$ and evaluated as complete heuristics on the training instances. Multi-objective survival selection then retains competitive and diverse architectures to form the initial population $\mathcal{P}_0$. Further details are provided in Appendix B.1.

2.3 Joint Module–Architecture Evolution

MOSAIC jointly evolves search modules and architectures through two complementary mechanisms. Module rewriting modifies the search behavior implemented by a module, while architecture variation changes how modules are composed and ordered within an architecture, as well as the architecture length.

Module Rewriting. At generation g, let $\mathcal{P}_g$ denote the current architecture population. Given a parent architecture $\mathcal{A} = (m_{i_1}, \ldots, m_{i_L}) \in \mathcal{P}_g$, MOSAIC selects a position j and invokes the LLM to rewrite the corresponding module m_{i_j} as

$$m'_{i_j} = \mathcal{R}(m_{i_j}), \qquad \mathcal{A}' = (m_{i_1}, \ldots, m'_{i_j}, \ldots, m_{i_L}), \tag{3}$$

where $\mathcal{R}$ denotes the module rewriting operation. MOSAIC uses two complementary rewriting strategies. Behavioral Rewriting modifies the core solution generation mechanism to induce a distinct search behavior, whereas Adaptive Refinement preserves the core mechanism while refining the existing search behavior.

Architecture Variation. Architecture variation changes how modules are organized within an architecture without modifying their executable code. Given two parent architectures $\mathcal{A}^{(1)} = (m_1^{(1)}, \ldots, m_{L_1}^{(1)})$ and $\mathcal{A}^{(2)} = (m_1^{(2)}, \ldots, m_{L_2}^{(2)})$, architecture crossover recombines their module sequences to generate an offspring architecture

$$\mathcal{A}' = (m_1^{(1)}, \ldots, m_{c_1}^{(1)}, m_{c_2+1}^{(2)}, \ldots, m_{L_2}^{(2)}), \tag{4}$$

where c_1 and c_2 denote the crossover positions in the two parent architectures. A structural mutation is then applied to $\mathcal{A}'$:

$$\mathcal{A}'' = \mathcal{S}_s(\mathcal{A}'), \qquad s \in \{\text{ins}, \text{del}, \text{rep}, \text{reord}\}, \tag{5}$$

where s denotes a feasible operation selected from insertion, deletion, replacement, and reordering. Together, crossover and structural mutation explore module composition, execution order, and architecture length.

Candidate architectures produced by module rewriting and architecture variation are evaluated as complete heuristics rather than by evaluating modules independently. Further details are provided in Appendix B.2.

2.4 Architecture Evaluation and Survival Selection

MOSAIC evaluates each architecture as a complete heuristic on the training instances within the underlying multi-objective optimization framework. Hypervolume (HV) (Shang & Ishibuchi, 2020) is used to measure the quality of the resulting solution sets.

Let the training instances be partitioned into T instance types $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$. For each architecture $\mathcal{A}$, MOSAIC first averages its HV over the training instances within each type, denoted by $H_t(\mathcal{A})$. Two architecture-level objectives are then defined as

$$F_{\text{macro}}(\mathcal{A}) = \frac{1}{T}\sum_{t=1}^{T} H_t(\mathcal{A}), \tag{6}$$

$$F_{\text{worst}}(\mathcal{A}) = \min_{t=1,\ldots,T} H_t(\mathcal{A}). \tag{7}$$

The former measures overall performance with equal weight assigned to each instance type, while the latter emphasizes performance on the weakest type. Together, they form the architecture objective vector $\mathbf{F}(\mathcal{A}) = (F_{\text{macro}}(\mathcal{A}), F_{\text{worst}}(\mathcal{A}))$.

Candidate architectures generated by module rewriting and architecture variation are merged with the current architecture population. MOSAIC then applies nondominated sorting and crowding distance based on $\mathbf{F}(\mathcal{A})$ to form the next population $\mathcal{P}_{g+1}$. The module pool is subsequently updated for the next generation. Further details are provided in Appendix B.3.

3 Experiments

3.1 Experimental Settings

Problems and Datasets. We evaluate MOSAIC on three bi-objective COPs: the bi-objective traveling salesman problem (Bi-TSP) (Paquete & Stützle, 2003), the bi-objective knapsack problem (Bi-KP) (Gandibleux & Freville, 2000), and the bi-objective green vehicle routing problem (Bi-GVRP) (Erdoğdu & Karabulut, 2022). Bi-TSP considers two distance objectives, Bi-KP maximizes two profit objectives under a capacity constraint, and Bi-GVRP jointly minimizes total travel distance and fuel consumption. For each problem, we use disjoint training and test sets. Training instances are used for initial module pool construction, architecture evolution, and final architecture selection, while test instances remain unseen throughout the search process and are used only for final evaluation. Detailed instance types, sizes, and train–test splits are provided in Appendix C.1.

Baselines. We consider two categories of LLM-driven AHD baselines. **(a) Multi-operator design methods.** We compare against E2OC (Qiu et al., 2026) and G-LNS (Zhao et al., 2026). For E2OC, Bi-TSP and Bi-GVRP use the three operators adopted in its original setting: 2-opt, 3-opt, and Or-opt. For Bi-KP, whose binary solution representation is incompatible with these route-based operators, we use bitflip, blockflip, and bitswap. For G-LNS, all three problems retain its predefined Destroy–Repair structure, with the search behaviors adapted to the solution representation of each problem. **(b) Single-operator design methods.** We compare against EoH (Liu et al., 2024), FunSearch (Romera-Paredes et al., 2024), ReEvo (Ye et al., 2024), and MCTS-AHD (Zheng et al., 2025). Each method designs an offspring generation function that is integrated into NSGA-II as the search operator.

We use NSGA-II (Deb et al., 2002) and MOEA/D (Zhang & Li, 2007) as the underlying evolutionary frameworks. Within each framework, we evaluate both the initial and best searched configurations of E2OC and MOSAIC. For G-LNS, we evaluate the initial and best Destroy–Repair configurations within its multi-objective LNS framework. Comparing the initial and searched configurations assesses the improvement achieved through multi-operator search. Evaluating E2OC and MOSAIC under both NSGA-II and MOEA/D further examines whether these improvements are consistent across evolutionary frameworks.

Metrics. We use Hypervolume (HV) (Guerreiro et al., 2021), Inverted Generational Distance (IGD) (Ishibuchi et al., 2014), and the relative HV gap for evaluation. The HV gap is computed for each instance as the difference between the reference HV and the mean HV over five independent runs, divided by the reference HV. All metrics are first averaged over instances within each type and then equally across types to avoid bias from unequal numbers of instances.

Implementation and Runtime Environment. All methods use the same termination criterion of 253,000 objective evaluations per training instance. For MOSAIC, the initial module pool and architecture population both have a size of 20, with a maximum architecture length of 4. All LLM-driven methods use DeepSeek-V4-Flash with a temperature of 0.7. Experiments are conducted on a 10-core Apple M5 CPU with 24 GB of unified memory. Detailed experimental settings are provided in Appendix C.2.

3.2 Main Results

Comparison with LLM-driven Multi-Operator Design Methods. Table 1 compares MOSAIC with E2OC and G-LNS from two perspectives. First, we assess the improvement achieved through search by comparing the initial designs with the best searched designs on the test sets. G-LNS yields substantial improvements on Bi-TSP and Bi-KP but no clear gain on Bi-GVRP, while E2OC achieves substantial improvements on all three problems. MOSAIC consistently improves over its initial architectures on all three problems, reducing the Gap by 70.7–85.4, 21.5–33.3, and 38.6–52.4 percentage points under NSGA-II on Bi-TSP, Bi-KP, and Bi-GVRP, respectively. Improvements are also observed under MOEA/D, indicating consistent gains across the two evolutionary frameworks.

Second, we evaluate cross-scale generalization. E2OC performance degrades as the problem scale increases on Bi-TSP and Bi-KP, while G-LNS maintains relatively stable performance across test scales. In contrast, MOSAIC maintains strong performance across test scales on all three problems under both NSGA-II and MOEA/D, demonstrating consistent cross-scale generalization. Mean and standard deviation results are reported in Appendix C.3.

Comparison with LLM-driven Single-Operator Design Methods. Table 2 compares MOSAIC with four single-operator design methods under the same NSGA-II framework. On Bi-TSP, all methods generalize well across problem scales, while MOSAIC consistently achieves better test performance. On Bi-KP, performance degrades as the problem scale increases for all methods, but MOSAIC maintains a clear advantage across all test scales. On Bi-GVRP, MOSAIC substantially outperforms the single-operator methods during training and maintains this advantage across test scales. Overall, MOSAIC consistently outperforms the single-operator methods across all three problems, with the largest advantage observed on Bi-GVRP. These results support the benefit of extending heuristic design beyond a single-operator structure to the composition of multiple search behaviors. Mean and standard deviation results are reported in Appendix C.4.

Table 1: Comparison of MOSAIC with LLM-driven multi-operator design methods on Bi-TSP, Bi-KP, and Bi-GVRP. A_{init} and A_{best} denote the initial designs and the best designs selected during search on the training instances, respectively. Ref. PF denotes the best-known reference Pareto Front. Best and second-best test results among the searched designs are shown in bold and underlined, respectively. Results are averaged over five independent runs.

	Train	Test														
		Bi-TSP100			Bi-TSP150			Bi-TSP200			Bi-TSP300			Bi-TSP500		
Method	HV↑	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓
Ref. PF	-	0.91	0.0%	0.00	0.92	0.0%	0.00	0.93	0.0%	0.00	0.94	0.0%	0.00	0.96	0.0%	0.00
$A_{\text{init}}^{\text{G-LNS}}$ + LNS	–	0.66	27.4%	0.21	0.65	29.6%	0.23	0.68	26.7%	0.22	0.64	31.8%	0.27	0.67	29.9%	0.27
$A_{\text{best}}^{\text{G-LNS}}$ + LNS	0.70	0.72	21.3%	0.19	0.71	22.8%	0.22	0.72	22.0%	0.21	0.70	26.3%	0.25	0.72	24.9%	0.25
$A_{\text{init}}^{\text{E2OC}}$ + NSGA-II	–	0.18	80.0%	0.61	0.12	87.2%	0.72	0.10	89.4%	0.77	0.06	93.2%	0.86	0.05	95.3%	0.92
$A_{\text{init}}^{\text{E2OC}}$ + MOEA/D	–	0.19	79.2%	0.60	0.12	87.3%	0.72	0.11	89.2%	0.76	0.06	93.2%	0.86	0.05	95.0%	0.93
$A_{\text{best}}^{\text{E2OC}}$ + NSGA-II	0.67	0.66	27.7%	0.24	0.62	33.0%	0.32	0.60	35.9%	0.37	0.57	39.6%	0.45	0.55	42.8%	0.51
$A_{\text{best}}^{\text{E2OC}}$ + MOEA/D	0.66	0.66	27.6%	0.24	0.61	33.6%	0.32	0.58	37.6%	0.38	0.57	39.5%	0.45	0.56	41.9%	0.50
$A_{\text{init}}^{\text{MOSAIC}}$ + NSGA-II	–	0.20	77.5%	0.58	0.16	82.8%	0.66	0.13	85.8%	0.71	0.10	89.3%	0.78	0.08	91.9%	0.84
$A_{\text{init}}^{\text{MOSAIC}}$ + MOEA/D	–	0.33	64.2%	0.46	0.30	67.4%	0.51	0.27	70.6%	0.55	0.26	73.0%	0.63	0.24	74.6%	0.67
$A_{\text{best}}^{\text{MOSAIC}}$ + NSGA-II	0.80	**0.85**	**6.8%**	**0.05**	**0.86**	**6.6%**	**0.06**	**0.86**	**6.8%**	**0.06**	**0.88**	**7.2%**	**0.08**	**0.89**	**6.5%**	**0.08**
$A_{\text{best}}^{\text{MOSAIC}}$ + MOEA/D	0.76	0.82	10.0%	0.08	0.83	9.8%	0.09	0.83	10.0%	0.08	0.85	10.2%	0.10	0.87	9.2%	0.11

	Train	Test									Train	Test					
		Bi-KP200–300			Bi-KP350–400			Bi-KP450–500				Bi-GVRP100–199			Bi-GVRP200–483		
Method	HV↑	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓
Ref. PF	-	0.84	0.0%	0.00	0.85	0.0%	0.00	0.84	0.0%	0.00	-	0.80	0.0%	0.00	0.84	0.0%	0.00
$A_{\text{init}}^{\text{G-LNS}}$ + LNS	-	0.00	99.7%	1.35	0.00	99.4%	1.41	0.00	100.0%	1.88	-	0.22	73.2%	0.62	0.23	73.0%	0.64
$A_{\text{best}}^{\text{G-LNS}}$ + LNS	0.76	0.52	38.7%	0.29	0.50	40.9%	0.31	0.49	41.9%	0.31	0.24	0.22	73.3%	0.63	0.23	73.2%	0.64
$A_{\text{init}}^{\text{E2OC}}$ + NSGA-II	-	0.04	95.4%	0.97	0.02	97.7%	1.12	0.00	99.9%	1.52	-	0.28	64.6%	0.52	0.25	70.0%	0.60
$A_{\text{init}}^{\text{E2OC}}$ + MOEA/D	-	0.04	95.0%	0.95	0.02	97.6%	1.08	0.00	99.9%	1.49	-	0.30	62.8%	0.50	0.26	69.6%	0.59
$A_{\text{best}}^{\text{E2OC}}$ + NSGA-II	0.60	0.51	39.5%	0.26	0.48	43.7%	0.31	0.41	50.9%	0.40	0.47	0.48	40.4%	0.30	0.64	24.8%	0.19
$A_{\text{best}}^{\text{E2OC}}$ + MOEA/D	0.61	0.51	39.5%	0.26	0.48	43.9%	0.31	0.41	50.7%	0.40	0.47	0.48	40.6%	0.30	0.64	24.8%	0.19
$A_{\text{init}}^{\text{MOSAIC}}$ + NSGA-II	-	0.42	49.9%	0.37	0.37	55.8%	0.43	0.29	65.4%	0.57	-	0.36	55.3%	0.43	0.32	61.8%	0.51
$A_{\text{init}}^{\text{MOSAIC}}$ + MOEA/D	-	0.60	28.7%	0.21	0.58	30.9%	0.22	0.54	36.0%	0.23	-	0.44	44.9%	0.35	0.50	40.3%	0.32
$A_{\text{best}}^{\text{MOSAIC}}$ + NSGA-II	0.61	**0.60**	**28.4%**	**0.21**	**0.60**	30.0%	**0.22**	**0.57**	32.1%	**0.22**	0.61	**0.67**	**16.7%**	**0.12**	0.76	**9.4%**	**0.06**
$A_{\text{best}}^{\text{MOSAIC}}$ + MOEA/D	0.61	**0.60**	28.7%	**0.21**	**0.60**	**29.9%**	**0.22**	**0.57**	**31.9%**	**0.22**	0.59	0.63	21.5%	0.16	**0.77**	**9.4%**	**0.06**

Statistical Significance Analysis. We conduct two-sided Wilcoxon signed-rank tests (Wilcoxon, 1945) within each problem size group. Test instances serve as paired statistical units, with performance averaged over five independent runs. MOSAIC is compared with each baseline within each group, and Holm correction (Holm, 1979) is applied across baseline comparisons, separately for HV and IGD. Problem size groups containing only a single test instance are excluded from statistical testing. Accordingly, three of the five Bi-TSP groups are tested, while all Bi-KP and Bi-GVRP groups are tested.

Compared with the multi-operator design methods, MOSAIC significantly outperforms the baselines on all three tested Bi-TSP groups, all three Bi-KP groups, and one of the two Bi-GVRP groups. Compared with the single-operator design methods, MOSAIC significantly outperforms the baselines on all three tested Bi-TSP groups and all three Bi-KP groups, while no significant improvement is observed on Bi-GVRP. Across both comparisons, no tested group shows significant underperformance by MOSAIC. IGD exhibits significance patterns consistent with those of HV. Detailed adjusted p-values are reported in Appendix C.5.

3.3 Ablation Studies

We evaluate two key components of MOSAIC: module rewriting and architecture evolution. For module rewriting, we ablate Adaptive Refinement (AR), Behavioral Rewriting (BR), and both mechanisms to assess their individual and combined contributions. We further introduce Fixed-Arch, which keeps the architecture fixed during evolution while retaining module rewriting, to evaluate the contribution of evolving module organization. All experiments follow the same protocol and parameter settings as the main experiments.

Table 2: Comparison of MOSAIC with LLM-driven single-operator design methods under NSGA-II on Bi-TSP, Bi-KP, and Bi-GVRP. Ref. PF denotes the best-known reference Pareto Front. Best and second-best test results are shown in bold and underlined, respectively. Results are averaged over five independent runs.

	Train	Test														
		Bi-TSP100			Bi-TSP150			Bi-TSP200			Bi-TSP300			Bi-TSP500		
Method	HV↑	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓
Ref. PF	-	0.91	0.0%	0.00	0.92	0.0%	0.00	0.93	0.0%	0.00	0.94	0.0%	0.00	0.96	0.0%	0.00
EoH	0.72	0.75	17.4%	0.16	0.69	25.5%	0.24	0.77	17.2%	0.17	0.79	16.2%	0.21	0.82	14.7%	0.21
FunSearch	0.78	0.83	8.7%	0.08	0.77	16.3%	0.15	0.85	8.8%	0.09	0.86	9.0%	0.13	0.88	8.2%	0.14
ReEvo	0.78	0.82	9.9%	0.09	0.84	8.6%	0.09	0.84	9.2%	0.10	0.86	9.2%	0.14	0.87	8.5%	0.15
MCTS-AHD	0.78	0.82	9.7%	0.09	0.84	8.9%	0.10	0.84	9.8%	0.10	0.86	9.3%	0.13	0.87	8.6%	0.14
MOSAIC	0.80	**0.85**	**6.8%**	**0.05**	**0.86**	**6.6%**	**0.06**	**0.86**	**6.8%**	**0.06**	**0.88**	**7.2%**	**0.08**	**0.89**	**6.5%**	**0.08**

	Train	Test									Train	Test					
		Bi-KP200–300			Bi-KP350–400			Bi-KP450–500				Bi-GVRP100–199			Bi-GVRP200–483		
Method	HV↑	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓	HV↑	Gap↓	IGD↓
Ref. PF	-	0.84	0.0%	0.00	0.85	0.0%	0.00	0.84	0.0%	0.00	-	0.80	0.0%	0.00	0.84	0.0%	0.00
EoH	0.56	0.58	31.2%	0.24	0.58	32.3%	0.25	0.55	33.9%	0.25	0.51	0.54	32.5%	0.24	0.73	13.3%	0.09
FunSearch	0.58	0.59	30.2%	0.22	0.58	31.4%	0.24	0.56	33.2%	0.24	0.51	0.53	33.5%	0.25	0.73	13.4%	0.09
ReEvo	0.57	0.58	30.7%	0.23	0.58	31.8%	0.25	0.56	33.6%	0.24	0.51	0.53	33.7%	0.25	0.73	13.4%	0.09
MCTS-AHD	0.58	0.59	30.5%	0.22	0.58	31.3%	0.24	0.56	33.6%	0.24	0.51	0.54	33.4%	0.25	0.73	13.4%	0.09
MOSAIC	0.61	**0.60**	**28.4%**	**0.21**	**0.60**	**30.0%**	**0.22**	**0.57**	**32.1%**	**0.22**	0.61	**0.67**	**16.7%**	**0.12**	**0.76**	**9.4%**	**0.06**

Table 3: Statistical significance of MOSAIC against the baselines across problem size groups. W/T/L denotes the numbers of tested size groups in which MOSAIC shows a significant win, a non-significant comparison, or a significant loss, respectively.

	Bi-TSP (W/T/L)		Bi-KP (W/T/L)		Bi-GVRP (W/T/L)	
Method	HV	IGD	HV	IGD	HV	IGD
EoH	3/0/0	3/0/0	3/0/0	3/0/0	0/2/0	0/2/0
FunSearch	3/0/0	3/0/0	3/0/0	3/0/0	0/2/0	0/2/0
ReEvo	3/0/0	3/0/0	3/0/0	3/0/0	0/2/0	0/2/0
MCTS-AHD	3/0/0	3/0/0	3/0/0	3/0/0	0/2/0	0/2/0
G-LNS	3/0/0	3/0/0	3/0/0	3/0/0	1/1/0	1/1/0
E2OC	3/0/0	3/0/0	3/0/0	3/0/0	1/1/0	1/1/0

As shown in Table 4, ablating either AR or BR degrades performance, while removing both leads to a larger decline across all three problems. This indicates that both mechanisms contribute to module evolution. Fixed-Arch also consistently degrades performance compared with the complete MOSAIC, indicating that evolving the architecture provides additional benefits beyond optimizing module behaviors alone. Overall, the complete MOSAIC achieves the best performance across all three problems, supporting the effectiveness of joint module–architecture evolution. Detailed means and standard deviations are reported in Appendix C.6.

3.4 Analysis of LLM Choice and Usage

Dependence on LLM Selection. We evaluate MOSAIC with four LLMs: DeepSeek-V4 Flash, DeepSeek-V4 Pro, GLM-5.3, and GPT-5.6 Sol, using the same settings as in the main experiments. As shown in Table 5, MOSAIC maintains strong performance across all four LLMs. GPT-5.6 Sol achieves the best overall performance across the three problems, followed by GLM-5.3. These results indicate that MOSAIC is not strongly dependent on the choice of LLM. Considering both performance and API cost, we use DeepSeek-V4 Flash as the default LLM in our experiments. Detailed means and standard deviations are reported in Appendix C.7.

LLM Usage and Generation Efficiency. To assess whether the performance gains of MOSAIC rely on increased LLM usage, we report the number of LLM API calls, the valid code generation

Table 4: Ablation results of MOSAIC under NSGA-II on Bi-TSP, Bi-KP, and Bi-GVRP. Best and second-best values are shown in bold and underlined, respectively. Results are averaged over five independent runs.

	Bi-TSP				Bi-KP				Bi-GVRP			
	Train	All Test			Train	All Test			Train	All Test		
Variant	HV↑	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓
w/o AR	<u>0.760</u>	<u>0.830</u>	<u>12.1%</u>	<u>0.104</u>	<u>0.590</u>	0.570	32.0%	0.230	<u>0.510</u>	<u>0.630</u>	<u>18.0%</u>	<u>0.170</u>
w/o BR	0.740	0.796	14.6%	0.111	<u>0.590</u>	<u>0.580</u>	<u>31.9%</u>	<u>0.220</u>	0.400	0.560	27.5%	0.230
w/o AR+BR	0.310	0.352	64.0%	0.462	0.240	0.180	79.2%	0.620	0.240	0.330	62.5%	0.470
Fixed-Arch	0.620	0.699	27.4%	0.237	0.580	0.570	32.0%	0.240	0.280	0.430	51.8%	0.360
MOSAIC	**0.798**	**0.870**	**6.9%**	**0.069**	**0.610**	**0.591**	**29.8%**	**0.217**	**0.610**	**0.715**	**11.0%**	**0.091**

Table 5: Comparison of MOSAIC with different LLMs under NSGA-II on Bi-TSP, Bi-KP, and Bi-GVRP. Best and second-best values are shown in bold and underlined, respectively. Results are averaged over five independent runs.

	Bi-TSP				Bi-KP				Bi-GVRP			
	Train	All Test			Train	All Test			Train	All Test		
LLM	HV↑	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓
DeepSeek-v4 Flash	0.798	0.870	6.9%	0.069	0.608	0.591	29.8%	0.217	0.610	0.715	11.0%	0.091
DeepSeek-v4 Pro	0.800	0.873	6.7%	0.067	<u>0.611</u>	0.594	29.6%	0.215	0.611	0.717	10.7%	**0.087**
GLM-5.3	<u>0.803</u>	<u>0.876</u>	<u>6.4%</u>	<u>0.064</u>	<u>0.611</u>	<u>0.595</u>	<u>29.4%</u>	<u>0.213</u>	<u>0.614</u>	<u>0.719</u>	<u>10.5%</u>	**0.087**
GPT-5.6 Sol	**0.809**	**0.879**	**6.0%**	**0.060**	**0.615**	**0.598**	**29.1%**	**0.210**	**0.617**	**0.722**	**10.2%**	**0.087**

rate, and the input and output token counts for each method. All methods use the same evaluation budget as in the main experiments, with up to three retries for invalid code generation. As shown in Table 6, MOSAIC requires fewer API calls and input and output tokens than all baselines except G-LNS across all three problems, while maintaining a high valid code generation rate. These results indicate that the performance gains of MOSAIC are not driven by greater LLM usage.

4 Discussion and Limitations

4.1 Module Synergy in Architecture Evolution

To examine how architecture evolution enables synergy among modules, we analyze the best architecture obtained during training on Bi-KP, $O_{782} \rightarrow O_{181} \rightarrow O_{103} \rightarrow O_{181}$. According to their executable code, O_{782} performs cross-individual information recombination, O_{181} applies population-guided adaptive mutation, and O_{103} additionally preserves high-quality parents during population-guided mutation. Their sequential organization combines information recombination, adaptive mutation, and elite preservation, suggesting complementary search behaviors that can work synergistically within the architecture. Architectures and module code for all three problems are provided in Appendix D.1.

To assess the role of execution order in this synergy, we keep the modules and their executable code unchanged and evaluate all 12 unique orderings of the four-module architecture. The architecture obtained during training achieves an HV of 0.585 and ranks second, close to the best ordering at 0.590 and above the average HV of 0.549 across the other 11 orderings. Moreover, all three orderings with O_{782} placed first rank among the top five, whereas all three with O_{782} placed last rank among the bottom three. IGD shows a consistent trend. Complete results for all execution orders are reported in Table 22 in Appendix D.1. Given the information recombination performed by O_{782}, the results suggest that applying information recombination before subsequent population-guided mutations leads to more effective search. More generally, the performance variation across different orderings shows that module synergy depends not only on the constituent search behaviors, but also on their execution order. This highlights the role of architecture evolution in organizing complementary module behaviors.

Table 6: Comparison of LLM usage and generation efficiency on Bi-TSP, Bi-KP, and Bi-GVRP. Calls denotes the number of LLM API calls, Valid denotes the valid code generation rate, and Input/Output denotes the input/output token counts in millions (M). Best and second-best values are shown in bold and underlined, respectively. Results are averaged over five independent runs.

	Bi-TSP				Bi-KP				Bi-GVRP			
Method	Calls ↓	Valid ↑	Input ↓	Output ↓	Calls ↓	Valid ↑	Input ↓	Output ↓	Calls ↓	Valid ↑	Input ↓	Output ↓
EoH	2,302	<u>89.4%</u>	9.35M	5.36M	2,304	90.2%	7.29M	4.00M	2,301	75.8%	8.67M	5.28M
FunSearch	2,869	80.2%	17.46M	10.54M	2,913	85.7%	13.29M	8.41M	2,965	71.2%	20.36M	14.12M
ReEvo	3,903	**99.8%**	10.63M	2.82M	3,890	**97.6%**	11.16M	3.08M	3,931	**96.1%**	11.54M	3.68M
MCTS-AHD	5,517	69.9%	11.19M	15.39M	5,663	76.2%	10.17M	13.24M	5,873	66.4%	13.26M	17.41M
E2OC	2,875	81.6%	3.41M	2.35M	3,234	83.3%	1.76M	1.53M	3,427	76.4%	2.73M	2.22M
G-LNS	**525**	82.7%	**0.18M**	**0.11M**	**600**	86.2%	**0.12M**	**0.22M**	**520**	74.2%	**0.17M**	**0.19M**
MOSAIC	<u>1,116</u>	84.6%	<u>2.15M</u>	<u>1.11M</u>	<u>1,117</u>	<u>92.2%</u>	<u>1.57M</u>	<u>0.83M</u>	<u>1,145</u>	<u>81.7%</u>	<u>2.34M</u>	<u>1.29M</u>

4.2 Architecture Search over Large Module Pools

Existing LLM-driven AHD methods typically rely on prompting strategies or design ideas to generate heuristic code. However, whether the generated code faithfully implements the intended guidance, and whether semantic drift occurs during generation, remain difficult to systematically verify. MOSAIC also relies on prompt-guided code generation for module rewriting and therefore does not fully avoid this issue. Section 4.1 shows that effective search can emerge from reorganizing fixed modules, suggesting a possible way to reduce this dependence. We therefore conduct a preliminary study on Bi-TSP by disabling module rewriting and performing only architecture evolution over different module pool sizes, with all other experimental settings identical to those used in the main experiments. Performance generally improves as the pool grows from 200 to 1,000 modules, approaching existing LLM-driven AHD methods at 1,000, but declines at 2,000. These results suggest the potential of architecture search over large executable module pools, while also revealing the increasing search difficulty under a fixed evaluation budget. Future work may investigate module selection, redundancy control, and scalable architecture search. Complete per-instance results are provided in Appendix D.2.

4.3 Initialization Robustness, Applicability, and Limitations

Under both NSGA-II and MOEA/D, architecture evolution consistently improves architectures from poor initial states, indicating limited sensitivity to initialization (see Appendix D.3). On the three-objective multi-objective flexible job-shop scheduling problem (MOFJSP), MOSAIC consistently achieves the best mean HV on test instances, supporting its broader applicability beyond the main benchmark problems (see Appendix D.4). We further analyze architecture length and sensitivity to the maximum architecture length. Although the maximum length is selected in most runs under the default setting, increasing the upper bound does not consistently improve performance. This suggests a trade-off between a larger architecture search space and increased search difficulty under a fixed evaluation budget (see Appendix D.5). Further limitations are discussed in Appendix D.6.

5 Conclusion

We propose MOSAIC, a modular search architecture evolution framework for LLM-driven AHD. MOSAIC represents heuristics as architectures composed of executable search modules and jointly explores search behaviors and their organization through module rewriting and architecture variation. Experiments across multiple MCOPs show that MOSAIC outperforms advanced LLM-driven AHD methods and exhibits strong cross-scale generalization. Further analysis provides evidence of order-dependent module synergy and shows that architecture evolution over a large module pool without module rewriting can approach the performance of existing LLM-driven AHD methods, suggesting a promising step toward heuristic design beyond repeated prompt-guided code generation. Future work will focus on more efficient module selection and architecture search for larger module pools. The source code is available at `https://anonymous.4open.science/r/Anonymous_code-A003/`.

AI USE STATEMENT

The research problem, MOSAIC framework, experimental protocol, implementation, experiments, analysis of the results, and manuscript preparation were conducted by the authors. LLMs were used as an integral component of MOSAIC and the LLM-driven baselines to generate candidate search modules or heuristic programs, as well as for limited assistance with code development, debugging, and language editing. All experimental results, analyses, and conclusions were verified by the authors, who take full responsibility for the final manuscript.

REFERENCES

Malti Baghel, Shikha Agrawal, and Sanjay Silakari. Survey of metaheuristic algorithms for combinatorial optimization. *International Journal of Computer Applications*, 58(19):21–31, 2012.

Aymeric Blot, Marie-Éléonore Kessaci, and Laetitia Jourdan. Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation: A. blot et al. *Journal of Heuristics*, 24(6):853–877, 2018.

Thomas Bömer, Bastian Amberg, Max Disselnmeyer, and Anne Meyer. Coupleevo: Evolving heuristics for coupled optimization problems using large language models. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1353–1361, 2026.

Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.

Edmund K Burke, Matthew R Hyde, Graham Kendall, and John Woodward. Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 1559–1565, 2007.

Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*, pp. 449–468. Springer, 2010.

Edmund K Burke, Matthew R Hyde, Graham Kendall, and John Woodward. Automating the packing heuristic design process with genetic programming. *Evolutionary computation*, 20(1):63–89, 2012.

Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

Christian L Camacho-Villalón, Thomas Stützle, and Marco Dorigo. Designing new metaheuristics: manual versus automatic approaches. *Intelligent Computing*, 2:0048, 2023.

Chentong Chen, Mengyuan Zhong, Ye Fan, Jialong Shi, and Jianyong Sun. Hifo-prompt: Prompting with hindsight and foresight for llm-based automatic heuristic design. In *International Conference on Learning Representations*, 2026.

Augusto B Corrêa, André G Pereira, and Jendrik Seipp. Classical planning with llm-generated heuristics: Challenging the state of the art with python code. *Advances in Neural Information Processing Systems*, 38:42070–42108, 2025.

Pham Vu Tuan Dat, Long Doan, and Huynh Thi Thanh Binh. Hsevo: Elevating automatic heuristic design with diversity-driven harmony search and genetic algorithm using llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26931–26938, 2025.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.

Mădălina M Drugan and Dirk Thierens. Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies. *Journal of Heuristics*, 18(5):727–766, 2012.

Matthias Ehrgott and Xavier Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Top*, 12(1):1–63, 2004.

Kazım Erdoğdu and Korhan Karabulut. Bi-objective green vehicle routing problem. *International Transactions in Operational Research*, 29(3):1602–1626, 2022.

Xavier Gandibleux and Arnaud Freville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objectives case. *Journal of Heuristics*, 6(3):361–383, 2000.

Xiaoyuan Guan, Tianyi Yang, Chunliang Zhao, and Yuren Zhou. Feedback-based adaptive crossover-rate in evolutionary computation. In *IJCAI*, pp. 6923–6930, 2024.

Andreia P Guerreiro, Carlos M Fonseca, and Luís Paquete. The hypervolume indicator: Computational problems and algorithms. *ACM Computing Surveys (CSUR)*, 54(6):1–42, 2021.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Ziyao Huang, Weiwei Wu, Kui Wu, Wei-Bin Lee, and Jianping Wang. Calm: Co-evolution of algorithms and language model for automatic heuristic design. In *International Conference on Learning Representations*, pp. 72468–72510, 2026.

Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems. In *2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pp. 170–177. IEEE, 2014.

Angel A Juan, Javier Faulin, Scott E Grasman, Markus Rabe, and Gonçalo Figueira. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72, 2015.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32201–32223, 2024.

Fei Liu, Yilu Liu, Qingfu Zhang, Tong Xialiang, and Mingxuan Yuan. Eoh-s: Evolution of heuristic set using llms for automated heuristic design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 37090–37098, 2026a.

Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Zhe Zhao, Xialiang Tong, Kun Mao, Zhichao Lu, Zhenkun Wang, et al. A systematic survey on large language models for algorithm design. *ACM Computing Surveys*, 58(8):1–32, 2026b.

Fei Liu, Rui Zhang, Shunyu Yao, Qinglong Hu, Zhichao Lu, Qingfu Zhang, et al. Hierarchical representations for cross-task automated heuristic design using llms. In *Forty-third International Conference on Machine Learning*, 2026c.

Yuxin Liu, Yi Mei, Mengjie Zhang, and Zili Zhang. Automated heuristic design using genetic programming hyper-heuristic for uncertain capacitated arc routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 290–297, 2017.

Thibaut Lust and Jacques Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In *Advances in Multi-Objective Nature Inspired Computing*, pp. 119–141. Springer, 2010.

Su Nguyen, Yi Mei, and Mengjie Zhang. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, 3(1):41–66, 2017.

Luis Paquete and Thomas Stützle. A two-phase local search for the biobjective traveling salesman problem. In *International conference on evolutionary multi-criterion optimization*, pp. 479–493. Springer, 2003.

Junhao Qiu, Xin Chen, Liang Ge, Liyong Lin, Zhichao Lu, and Qingfu Zhang. Evolving interdependent operators with large language models for multi-objective combinatorial optimization. *Proceedings of the International Conference on Machine Learning*, 2026.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Ke Shang and Hisao Ishibuchi. A new hypervolume-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 24(5):839–852, 2020.

Shengming Sun and Jialong Shi. Breaking validity-induced boundaries to expand algorithm search space: A two-stage ast-based operator for llm-driven automated heuristic evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1434–1440, 2026.

Ye Tian, Xiaopeng Li, Haiping Ma, Xingyi Zhang, Kay Chen Tan, and Yaochu Jin. Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(4):1051–1064, 2022.

Niki Van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29 (2):331–345, 2025.

Hui Wang, Yang Liu, Xiaoyu Zhang, and Chaoxu Mu. Cogmcts: A novel cognitive-guided monte carlo tree search framework for iterative heuristic evolution with large language models. *arXiv preprint arXiv:2512.08609*, 2025a.

Hui Wang, Xufeng Zhang, and Chaoxu Mu. Planning of heuristics: Strategic planning on large language models with monte carlo tree search for automating heuristic optimization. *arXiv preprint arXiv:2502.11422*, 2025b.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

Zhuoliang Xie, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Llm-driven neighborhood search for efficient heuristic design. In *2025 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2025.

Zhenxing Xu, Yizhe Zhang, Weidong Bao, Hao Wang, Ming Chen, Haoran Ye, Wenzheng Jiang, Hui Yan, and Ji Wang. Autoep: Llms-driven automation of hyperparameter evolution for metaheuristic algorithms. In *International Conference on Learning Representations*, pp. 139927–139945, 2026.

Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 27144–27152, 2025.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

Baoyun Zhao, He Wang, and Liang Zeng. G-lns: Generative large neighborhood search for llm-based automatic heuristic design. *arXiv preprint arXiv:2602.08253*, 2026.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in llm-based automatic heuristic design. *In Proceedings of the International Conference on Machine Learning*, 2025.

APPENDIX CONTENTS

A Related Works **14**

A.1 Multi-objective Combinatorial Optimization 14

A.2 Automated Heuristic Design 14

A.3 LLM-Driven Automated Heuristic Design 14

B More Method Details **15**

B.1 Module Pool and Architecture Initialization 15

B.2 Joint Module–Architecture Evolution 17

B.3 Architecture Evaluation and Survival Selection 18

C Additional Experimental Details **19**

C.1 Instance Sizes and Train–Test Splits 19

C.2 Detailed Experimental Settings 19

C.3 Comparison Results with Multi-Operator Design Methods 20

C.4 Comparison Results with Single-Operator Design Methods 20

C.5 Statistical Significance Results 27

C.6 Detailed Ablation Results 27

C.7 Detailed Results across Different LLMs 30

D Additional Discussion and Analysis **34**

D.1 Additional Details on Module Synergy 34

D.2 Additional Results on Large Module Pools 34

D.3 Additional Results on Initialization Sensitivity 44

D.4 Additional Experiments on MOFJSP 44

D.5 Analysis of Architecture Length 45

D.6 Limitations 45

E Licenses **46**

A Related Works

A.1 Multi-objective Combinatorial Optimization

Multi-objective combinatorial optimization problems (MCOPs) optimize multiple conflicting objectives over discrete solution spaces and arise in applications such as routing, scheduling, and resource allocation (Ehrgott & Gandibleux, 2004). Due to their computational complexity, multi-objective evolutionary algorithms and metaheuristics are widely used to approximate Pareto-optimal solution sets (Lust & Teghem, 2010). Representative frameworks include NSGA-II (Deb et al., 2002) and MOEA/D (Zhang & Li, 2007).

Beyond general frameworks, extensive work has developed problem-specific search operators for MCOPs, including crossover, mutation, neighborhood search, and local improvement mechanisms (Blot et al., 2018). Multiple search behaviors are also often combined within a solver, for example by integrating evolutionary and local search or coordinating different neighborhoods (Drugan & Thierens, 2012). These studies show that both individual search behaviors and their coordination can substantially affect optimization performance.

However, such mechanisms are typically designed manually for individual problems and rely on substantial domain knowledge, motivating research on automated heuristic design.

A.2 Automated Heuristic Design

Automated heuristic design (AHD) searches over heuristic design spaces rather than directly over problem solution spaces, aiming to reduce manual algorithm design (Burke et al., 2010; 2013). Classical AHD broadly includes heuristic selection and heuristic generation. The former selects or combines low-level heuristics from a predefined set, while the latter constructs new heuristics from predefined primitives or components (Burke et al., 2013).

Genetic programming (GP) is a representative heuristic-generation approach (Nguyen et al., 2017). It represents candidate heuristics as evolvable program structures and has been applied to packing, routing, and scheduling (Burke et al., 2007; 2012; Liu et al., 2017). Component-based automated algorithm design similarly searches over configurations or combinations of predefined algorithmic components (Camacho-Villalón et al., 2023).

A key characteristic of these approaches is that the design space is specified in advance. GP relies on predefined function and terminal sets, while component-based methods require predefined component libraries and composition rules. Thus, component organization may be searched, but the available behavioral building blocks remain bounded by predefined primitives or components. In contrast, LLM-driven AHD can directly generate and rewrite executable heuristic code, allowing the available search behaviors themselves to evolve.

A.3 LLM-Driven Automated Heuristic Design

Recent advances in code generation have enabled large language models (LLMs) to act as heuristic designers within automated search processes. Existing LLM-driven AHD methods typically embed an LLM in an outer-loop search mechanism, where candidate heuristic programs are iteratively generated, evaluated, and revised according to performance feedback.

For multi-behavior heuristic design, two complementary design dimensions can be distinguished. The *behavior space* determines what search behavior is implemented by each heuristic component, while the *organization space* determines how multiple behaviors are composed and ordered within a complete search procedure. LLM-driven AHD has substantially expanded the behavior space through open-ended code generation and rewriting, whereas organization is often specified in advance.

Single-Operator Design. Most existing LLM-driven AHD methods treat an individual heuristic program or search operator as the primary design unit. FunSearch (Romera-Paredes et al., 2024), EoH (Liu et al., 2024), and LLaMEA (Van Stein & Bäck, 2025) adopt evolutionary program search with different candidate-generation and evolution mechanisms. ReEvo (Ye et al., 2024) uses performance feedback and reflection to guide heuristic evolution, while HiFo-Prompt (Chen et al., 2026)

leverages historical experience and the current search state. Recent AST-based methods further enlarge the program search space by performing structural crossover and mutation on heuristic code and using LLMs for program repair (Sun & Shi, 2026). These methods improve the exploration of individual search behaviors, but do not explicitly search the organization of multiple interacting behaviors.

Some studies consider multiple heuristics without organizing them as interacting operators within a single search process. EoH-S (Liu et al., 2026a) evolves complementary heuristics for different instance types, while CoupleEvo (Bömer et al., 2026) coordinates heuristics for multiple interdependent subproblems. These works extend LLM-driven AHD beyond a single heuristic, but the heuristics serve different instances or subproblems rather than forming an evolvable organization within one search process.

Multi-Operator Design. A more closely related line of work jointly designs multiple interacting search operators. E2OC (Qiu et al., 2026) predefines operator slots and their execution order, then searches for promising design-strategy combinations across slots. G-LNS (Zhao et al., 2026) adopts a fixed Destroy–Repair organization and separately generates and evolves the operators assigned to these two roles.

These methods demonstrate the benefit of jointly designing multiple LLM-generated search behaviors, but their organizational structures are specified in advance. Search determines the behaviors assigned to predefined roles rather than the roles and their organization themselves. This contrasts with classical component-based AHD, where organization may be searched but the available components are typically predefined. MOSAIC combines these two capabilities by jointly evolving executable search behaviors and their organization through module rewriting and architecture evolution. Accordingly, MOSAIC extends LLM-driven AHD from behavior optimization under predefined structures to joint behavior–organization evolution.

B More Method Details

B.1 Module Pool and Architecture Initialization

Module Interface. MOSAIC uses a unified module interface that allows independently generated search modules to be executed and composed within the underlying multi-objective optimization framework. At generation t of the underlying optimization process, the framework provides each module with a reference state

$$R_t = (P_t, F_t, B_t), \tag{8}$$

where P_t denotes the current population, F_t contains its objective values, and B_t denotes the set of solutions selected by the original parent selection mechanism. Each module processes the current set of candidate solutions and returns a new set of the same size.

Initial Module Generation and Validation. MOSAIC constructs an initialization prompt based on the problem specification and unified module interface and invokes the LLM to generate executable search modules. The prompt provides the information required for module execution while leaving the specific search behavior unspecified. Each generated module undergoes validity and duplication checks before evaluation. Valid and distinct modules are evaluated on the training instances, and the top-performing modules are retained to form the initial module pool $\mathcal{M}_0$. The prompts for Bi-TSP, Bi-KP, and Bi-GVRP are shown in Figures 2–4.

Architecture Initialization and Execution. Following Eq. (1), initial architectures are sampled from $\mathcal{H}(\mathcal{M}_0)$ with lengths bounded by $L_{\max}$. Modules are sampled from $\mathcal{M}_0$ with replacement, allowing the same module to appear at multiple positions within an architecture. Duplicate architectures are resampled.

Given an architecture $\mathcal{A} = (m_{i_1}, m_{i_2}, \ldots, m_{i_L})$, its modules are executed sequentially as

$$X_0 = \mathrm{Copy}(B_t), \qquad X_j = m_{i_j}(X_{j-1}, R_t, C), \quad j = 1, \ldots, L, \tag{9}$$

where C denotes the problem context. The output of each module serves as the input to the next module, and X_L denotes the final set of candidate solutions produced by the architecture. Each architecture is evaluated as a complete heuristic on the training instances. Multi-objective survival selection is then applied to form the initial architecture population $\mathcal{P}_0$.

Bi-TSP Initial Module Generation Prompt

Task Description

Create one executable state-transition module for the offspring pipeline of a fixed bi-objective TSP solver. Follow the supplied interface exactly. Return a new batch of legal city permutations with the same shape, and make the result depend on `candidates`, the current pipeline state. `reference` and `context` are read-only attribute objects; use only `rng` for randomness. `selected_indices` indexes `parent_population`, not `mating_pool`; sample a mating-pool member by its direct row index.

Do not evaluate objectives, perform environmental selection, run a private optimization/local-search loop, access files or networks, or rebuild the whole batch independently of `candidates`. This function is called repeatedly: avoid unbounded while-loops and deep nested loops, and prefer NumPy batch operations. No named crossover or mutation mechanism is required. Return one concise algorithm description and the complete `module` code.

Correctness checklist (any violation is rejected before scoring):

1. Use the parameter names exactly as written in the header: `candidates`, `reference`, `context`, `generation`, and `rng`.
2. Every code path must return a new NumPy array of the same shape.
3. Do not define helper functions or other code before `module`.
4. Randomness must come only from the passed `rng`.
5. Allowed imports inside the module: numpy, math, itertools, functools, collections, heapq, and operator.
6. No file, network, or process operations.

Create one new module from scratch.

Return exactly:

1. One sentence describing the transition, enclosed in {braces}.
2. The complete implementation of:

```
import numpy as np

def module (candidates, reference, context, generation, rng):
    """ The function body of the module behavior."""
    result = np.array(candidates, copy=True)
    return result
```

Do not include any other explanation.

Figure 2: Initial module generation prompt for Bi-TSP.

Bi-KP Initial Module Generation Prompt

Task Description

Create one executable state-transition module for the offspring pipeline of a fixed bi-objective 0-1 knapsack solver. Follow the supplied interface exactly. Return a new same-shaped binary batch that depends on `candidates`; do not rebuild it independently from zeros, reference, profits, weights, or capacities. `reference` and `context` are read-only attribute objects and `rng` is the only random source. The framework performs the common capacity repair after the module, so do not implement a private repair.

Do not evaluate objectives, perform environmental selection, run a private optimization/local-search loop, or access files or networks. This function is called repeatedly: avoid unbounded while-loops and deep nested loops, and prefer NumPy batch operations. Return one concise algorithm description and the complete `module` code.

Correctness checklist (any violation is rejected before scoring):

1. Use the parameter names exactly as written in the header: `candidates`, `reference`, `context`, `generation`, and `rng`.
2. Every code path must return a new NumPy array of the same shape.
3. Do not define helper functions or other code before `module`.
4. Randomness must come only from the passed `rng`.
5. Allowed imports inside the module: numpy, math, itertools, functools, collections, heapq, and operator.
6. No file, network, or process operations.

Create one new module from scratch.

Return exactly:

1. One sentence describing the transition, enclosed in {braces}.
2. The complete implementation of:

```
import numpy as np

def module (candidates, reference, context, generation, rng):
    """ The function body of the module behavior ."""
    result = np.array(candidates, copy=True)
    return result
```

Do not include any other explanation.

Figure 3: Initial module generation prompt for Bi-KP.

Bi-GVRP Initial Module Generation Prompt

Task Description

Create one executable state-transition module for the offspring pipeline of a fixed bi-objective GVRP solver. Follow the supplied interface exactly. Values below `customer_count` are customers; larger values are unique route-separator tokens. Return a new same-shaped legal permutation that depends on `candidates`. `reference` and `context` are read-only attribute objects; use only `rng` for randomness. `selected_indices` indexes `parent_population`, not `mating_pool`; sample a mating-pool member by its direct row index.

Do not evaluate objectives, perform environmental selection, run a private optimization/local-search loop, access files or networks, or rebuild the whole batch independently of `candidates`. This function is called repeatedly: avoid unbounded while-loops and deep nested loops, and prefer NumPy batch operations. No named crossover or mutation mechanism is required. Return one concise algorithm description and the complete `module` code.

Correctness checklist (any violation is rejected before scoring):

1. Use the parameter names exactly as written in the header: `candidates`, `reference`, `context`, `generation`, and `rng`.
2. Every code path must return a new NumPy array of the same shape.
3. Do not define helper functions or other code before `module`.
4. Randomness must come only from the passed `rng`.
5. Allowed imports inside the module: numpy, math, itertools, functools, collections, heapq, and operator.
6. No file, network, or process operations.

Create one new module from scratch.

Return exactly:

1. One sentence describing the transition, enclosed in {braces}.
2. The complete implementation of:

```
import numpy as np

def module (candidates, reference, context, generation, rng):
    """ The function body of the module behavior."""
    result = np.array(candidates, copy=True)
    return result
```

Do not include any other explanation.

Figure 4: Initial module generation prompt for Bi-GVRP.

B.2 Joint Module–Architecture Evolution

Module Rewriting. At generation g, MOSAIC selects a parent architecture from the current population $\mathcal{P}_g$ via binary tournament selection based on nondominated rank and crowding distance, and uniformly samples a module position for rewriting. MOSAIC employs two complementary rewriting prompts. Behavioral Rewriting modifies the core solution generation mechanism to explore distinct search behaviors, whereas Adaptive Refinement preserves the core mechanism while refining the existing search behavior. The two prompts are shared across the three problem domains and are shown in Figures 5.

The rewritten module undergoes the same validity and duplication checks described in Appendix B.1. A successful rewrite replaces the selected module as defined in Eq. (3). The rewritten module is not evaluated independently; instead, the resulting architecture is evaluated as a complete heuristic.

Architecture Crossover. Architecture variation operates only on module sequences and does not modify module executable code or invoke the LLM. Two parent architectures are selected from the current population via binary tournament selection. When crossover is applied, the offspring architecture is constructed according to Eq. (4) by concatenating a prefix of one parent with a suffix of the other. If crossover is not applied, one of the two parent architectures is copied.

Structural Mutation. Following crossover or parent copying, exactly one structural mutation is applied to the resulting architecture. Let $\mathcal{A} = (m_1, \ldots, m_L)$ denote the architecture before mutation. One operation is selected uniformly from the feasible subset of insertion, deletion, replacement, and reordering. Insertion adds a module from the modules retained in the current population at a randomly selected position when $L < L_{\max}$. Deletion removes a randomly selected module when $L > 1$. Replacement substitutes a randomly selected module with a different retained module while preserving the architecture length. Reordering exchanges two modules at different positions and is available when the architecture contains at least two distinct modules.

Behavioral Rewriting Prompt

{Task Prompt}

Adapt this selected module. Description:
{Parent Module Description}.

Code:
{Parent Module Code}.

Behavioral Rewriting: Modify the module's core candidate-generation mechanism to obtain a behaviorally distinct transition. The rewrite may change how candidate solutions are selected, transformed, combined, or locally modified, while preserving the prescribed module interface and validity requirements.

Return exactly:
1. One sentence describing it, enclosed in {braces}.
2. The complete implementation of:

{FUNCTION_TEMPLATE}

Do not include any other explanation.

Adaptive Refinement Prompt

{Task Prompt}

Adapt this selected module. Description:
{Parent Module Description}.

Code:
{Parent Module Code}.

Adaptive Refinement: Preserve the module's core candidate-generation mechanism. Modify its parameter settings, application intensity, sampling probabilities, or generation-dependent control to obtain a distinct adaptive variant.

Return exactly:
1. One sentence describing it, enclosed in {braces}.
2. The complete implementation of:

{FUNCTION_TEMPLATE}

Do not include any other explanation.

Figure 5: Two rewriting prompts for module rewriting.

Thus, each offspring architecture undergoes at most one crossover followed by exactly one structural mutation. Together, these operations modify module composition, execution order, or architecture length without changing module executable code.

Module Retention. After architecture evaluation and survival selection, modules contained in the surviving architectures are retained for subsequent evolution. Accordingly, a newly rewritten module can participate in subsequent architecture variation only if its associated architecture survives.

B.3 Architecture Evaluation and Survival Selection

Architecture Evaluation. Each candidate architecture is evaluated as a complete heuristic on the training instances within the underlying multi-objective optimization framework. The modules are executed sequentially according to Eq. (9), while the remaining optimization process follows the original framework. The resulting HV values are averaged within each instance type to obtain $H_t(\mathcal{A})$, from which the two architecture-level objectives are computed.

Survival Selection. At generation g, candidate architectures generated by module rewriting and architecture variation are merged with the current population $\mathcal{P}_g$. Architectures that violate the module interface, produce invalid solutions, or exceed the prescribed evaluation time limit are treated as infeasible and ranked below feasible architectures. MOSAIC then performs nondominated sorting based on $\mathbf{F}(\mathcal{A})$ and uses crowding distance to preserve diversity. The surviving architectures form $\mathcal{P}_{g+1}$, after which the module pool is updated for subsequent evolution.

After the final generation, $\mathcal{A}^\star$ is selected from the final population by prioritizing lower nondomination rank and higher crowding distance.

Overall Search Procedure. Algorithm 1 summarizes the complete MOSAIC procedure.

Algorithm 1 Overall Procedure of MOSAIC

Require: Training instances $\mathcal{D}_{\text{train}}$; architecture population size N_{arch}; maximum architecture length $L_{\max}$; per-instance design budget B_{FES}; maximum number of generations $G_{\max}$
Ensure: Selected architecture $\mathcal{A}^{\star}$

```
Construct the initial module pool M_0
Initialize the architecture population P_0
for g = 0, ..., G_max − 1 do
    if the design budget is exhausted then
        break
    end if
    Generate module rewriting offspring O_g^rew from P_g
    Generate architecture variation offspring O_g^var from P_g
    Evaluate the generated offspring on D_train
    Form Q_g = P_g ∪ O_g^rew ∪ O_g^var
    Apply survival selection to form P_{g+1}
    Update the module pool
end for
Select A* from the final population based on nondomination rank and crowding distance
return A*
```

Table 7: Instance sizes and train–test splits for the three problems.

Problem	Split	Instance type	Size	Per size	Total
Bi-TSP	Train	Euclidean / Random / Mixed	100 cities	3 each	9
	Test	Euclidean / Random / Mixed	300 cities	3 each	9
		Euclidean / Random / Mixed	500 cities	3 each	9
		kro	100 cities	10	10
		kro	150 cities	1	1
		kro	200 cities	1	1
Bi-KP	Train	Xavier (A–D)	50, 100, 150 items	4 each	12
	Test	Xavier (A–D)	200–500 (step 50) items	4 each	28
Bi-GVRP	Train	CMT	50–100 customers	–	3
		Golden	200–400 customers	–	6
	Test	CMT	100–199 customers	–	4
		Golden	200–483 customers	–	14

C Additional Experimental Details

C.1 Instance Sizes and Train–Test Splits

Table 7 summarizes the instance types, sizes, and train–test splits for the three problems. Bi-TSP includes Euclidean, Random, Mixed, and kro instances. Bi-KP uses the Xavier instance family, whereas Bi-GVRP uses the CMT and Golden families from CVRPLIB. Some Golden instances additionally impose maximum travel-distance constraints. For type-wise aggregation of Bi-TSP results, kro instances are grouped with Euclidean instances.

C.2 Detailed Experimental Settings

All LLM-driven methods use DeepSeek-V4-Flash with a temperature of 0.7 and a maximum output length of 4,096 tokens. Each method is run independently five times with seeds $\{2026, 2027, 2028, 2029, 2030\}$. During training, candidate heuristics are evaluated with a population size of 10 for 10 generations and a 10-second time limit per instance. The search budget is fixed at 253,000 objective evaluations per training instance for all methods. Failed LLM generations are retried up to three times. During testing, all methods use a population size of 50 for 50 generations. Method-specific settings are summarized in Table 8.

Table 8: Method-specific parameter settings.

Method	Parameters
EoH	Program population $= 20$; selected programs $= 2$.
FunSearch	Samples per prompt $= 1$.
ReEvo	Program population $= 20$; mutation rate $= 0.5$.
MCTS-AHD	Initial program population $= 20$; maintained population $= 20$; selected programs $= 2$; $\alpha = 0.5$; $\lambda_0 = 0.1$.
E2OC	MCTS outer iterations $= 30$; simulations per iteration $= 1$; prompts per operator $= 4$; operator rotation interval $= 5$; internal EoH population $= 10$; generations $= 10$; maximum samples $= 25$.
G-LNS	Destroy pool size $= 5$; Repair pool size $= 5$; training archive size $= 10$; destroy ratio $= 0.2$; initial temperature $= 100$; temperature decay rate $= 0.97$; component evolution interval $= 10$.
MOSAIC	Initial module pool size $= 20$; architecture population size $= 20$; candidates per generation: rewriting $= 10$; architecture variation $= 10$; maximum architecture length $= 4$.

C.3 Comparison Results with Multi-Operator Design Methods

Tables 9–11 report detailed per-instance HV and IGD results for the multi-operator design methods on Bi-TSP, Bi-KP, and Bi-GVRP. Results are reported as the mean and standard deviation over five independent runs.

On Bi-TSP, MOSAIC achieves the best and second-best results on nearly all instances under NSGA-II and MOEA/D, respectively, with its advantage over E2OC becoming more pronounced on larger instances. On Bi-KP, MOSAIC generally achieves better HV and IGD across the tested scales, with particularly consistent advantages on larger instances. On Bi-GVRP, MOSAIC under both frameworks achieves the strongest performance on most instances, with clear improvements over G-LNS and E2OC. These results support the benefit of extending LLM-driven multi-operator AHD beyond predefined operator organizations by jointly evolving search behaviors and their organization.

C.4 Comparison Results with Single-Operator Design Methods

We compare MOSAIC with EoH, FunSearch, ReEvo, and MCTS-AHD under the same NSGA-II framework. Tables 12–14 report detailed per-instance HV and IGD results on Bi-TSP, Bi-KP, and Bi-GVRP. Results are reported as the mean and standard deviation over five independent runs.

On Bi-TSP, MOSAIC achieves strong performance across problem scales, attaining the best HV and IGD on nearly all instances. On Bi-KP, MOSAIC generally achieves the best or second-best performance across the tested scales, with consistent advantages over the single-operator baselines. On Bi-GVRP, MOSAIC achieves substantial improvements on many instances, particularly where the single-operator methods exhibit limited performance. These results support the benefit of extending LLM-driven AHD beyond a single search operator by composing multiple search behaviors within an evolved architecture.

Table 9: Per-instance performance comparison between MOSAIC and LLM-driven multi-operator design methods on the Bi-TSP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. The best and second-best mean values are highlighted in bold and underlined, respectively.

Instance	Metric	G-LNS	$A^{\mathrm{E2OC}}_{\mathrm{best}}$+NSGA-II	$A^{\mathrm{E2OC}}_{\mathrm{best}}$+MOEA/D	$A^{\mathrm{MOSAIC}}_{\mathrm{best}}$+NSGA-II	$A^{\mathrm{MOSAIC}}_{\mathrm{best}}$+MOEA/D
kroAB100	HV	0.726 ± 0.052	0.680 ± 0.013	0.690 ± 0.013	$\mathbf{0.857} \pm 0.041$	$\underline{0.825} \pm 0.056$
	IGD	0.207 ± 0.074	0.219 ± 0.014	0.212 ± 0.019	$\mathbf{0.062} \pm 0.052$	$\underline{0.094} \pm 0.070$
kroAC100	HV	0.710 ± 0.060	0.680 ± 0.014	0.689 ± 0.019	$\mathbf{0.852} \pm 0.044$	$\underline{0.821} \pm 0.056$
	IGD	0.189 ± 0.057	0.211 ± 0.018	0.212 ± 0.018	$\mathbf{0.053} \pm 0.047$	$\underline{0.079} \pm 0.060$
kroAD100	HV	0.693 ± 0.053	0.638 ± 0.010	0.628 ± 0.008	$\mathbf{0.837} \pm 0.046$	$\underline{0.806} \pm 0.057$
	IGD	0.188 ± 0.064	0.263 ± 0.011	0.268 ± 0.002	$\mathbf{0.052} \pm 0.049$	$\underline{0.079} \pm 0.060$
kroAE100	HV	0.726 ± 0.058	0.646 ± 0.013	0.648 ± 0.010	$\mathbf{0.853} \pm 0.045$	$\underline{0.825} \pm 0.054$
	IGD	0.188 ± 0.057	0.271 ± 0.025	0.262 ± 0.026	$\mathbf{0.055} \pm 0.054$	$\underline{0.077} \pm 0.061$
kroBC100	HV	0.723 ± 0.046	0.645 ± 0.008	0.652 ± 0.008	$\mathbf{0.842} \pm 0.048$	$\underline{0.818} \pm 0.049$
	IGD	0.182 ± 0.067	0.260 ± 0.013	0.267 ± 0.014	$\mathbf{0.054} \pm 0.054$	$\underline{0.076} \pm 0.062$
kroBD100	HV	0.698 ± 0.059	0.645 ± 0.009	0.648 ± 0.012	$\mathbf{0.837} \pm 0.040$	$\underline{0.806} \pm 0.055$
	IGD	0.194 ± 0.063	0.253 ± 0.008	0.251 ± 0.008	$\mathbf{0.049} \pm 0.042$	$\underline{0.078} \pm 0.068$
kroBE100	HV	0.713 ± 0.052	0.673 ± 0.014	0.657 ± 0.009	$\mathbf{0.846} \pm 0.044$	$\underline{0.817} \pm 0.049$
	IGD	0.200 ± 0.070	0.226 ± 0.024	0.226 ± 0.019	$\mathbf{0.054} \pm 0.049$	$\underline{0.080} \pm 0.065$
kroCD100	HV	0.718 ± 0.065	0.644 ± 0.024	0.645 ± 0.024	$\mathbf{0.850} \pm 0.044$	$\underline{0.820} \pm 0.061$
	IGD	0.200 ± 0.068	0.267 ± 0.042	0.263 ± 0.041	$\mathbf{0.056} \pm 0.050$	$\underline{0.086} \pm 0.072$
kroCE100	HV	0.721 ± 0.053	0.661 ± 0.012	0.662 ± 0.014	$\mathbf{0.850} \pm 0.045$	$\underline{0.823} \pm 0.056$
	IGD	0.213 ± 0.081	0.242 ± 0.018	0.230 ± 0.023	$\mathbf{0.057} \pm 0.057$	$\underline{0.088} \pm 0.075$
kroDE100	HV	0.726 ± 0.048	0.669 ± 0.012	0.671 ± 0.013	$\mathbf{0.848} \pm 0.045$	$\underline{0.822} \pm 0.046$
	IGD	0.173 ± 0.061	0.229 ± 0.017	0.233 ± 0.017	$\mathbf{0.049} \pm 0.047$	$\underline{0.072} \pm 0.057$
kroAB150	HV	0.710 ± 0.069	0.616 ± 0.006	0.611 ± 0.010	$\mathbf{0.859} \pm 0.043$	$\underline{0.830} \pm 0.064$
	IGD	0.221 ± 0.078	0.320 ± 0.017	0.319 ± 0.018	$\mathbf{0.056} \pm 0.055$	$\underline{0.087} \pm 0.074$
kroAB200	HV	0.722 ± 0.063	0.596 ± 0.008	0.580 ± 0.003	$\mathbf{0.863} \pm 0.046$	$\underline{0.834} \pm 0.067$
	IGD	0.210 ± 0.076	0.371 ± 0.018	0.381 ± 0.019	$\mathbf{0.058} \pm 0.053$	$\underline{0.084} \pm 0.071$
euclidAB300	HV	0.699 ± 0.063	0.561 ± 0.003	0.583 ± 0.002	$\mathbf{0.851} \pm 0.052$	$\underline{0.826} \pm 0.067$
	IGD	0.270 ± 0.101	0.420 ± 0.003	0.429 ± 0.009	$\mathbf{0.067} \pm 0.079$	$\underline{0.098} \pm 0.093$
euclidCD300	HV	0.704 ± 0.071	0.573 ± 0.003	0.567 ± 0.001	$\mathbf{0.854} \pm 0.049$	$\underline{0.827} \pm 0.068$
	IGD	0.236 ± 0.087	0.397 ± 0.007	0.393 ± 0.008	$\mathbf{0.059} \pm 0.063$	$\underline{0.088} \pm 0.081$
euclidEF300	HV	0.697 ± 0.068	0.550 ± 0.002	0.554 ± 0.004	$\mathbf{0.848} \pm 0.053$	$\underline{0.820} \pm 0.072$
	IGD	0.251 ± 0.096	0.452 ± 0.005	0.465 ± 0.001	$\mathbf{0.065} \pm 0.073$	$\underline{0.093} \pm 0.091$
mixedGG300	HV	0.686 ± 0.067	0.568 ± 0.003	0.552 ± 0.006	$\mathbf{0.882} \pm 0.042$	$\underline{0.851} \pm 0.063$
	IGD	0.274 ± 0.086	0.451 ± 0.010	0.444 ± 0.001	$\mathbf{0.082} \pm 0.078$	$\underline{0.109} \pm 0.082$
mixedHH300	HV	0.686 ± 0.056	0.570 ± 0.003	0.563 ± 0.001	$\mathbf{0.876} \pm 0.047$	$\underline{0.850} \pm 0.060$
	IGD	0.285 ± 0.099	0.447 ± 0.007	0.452 ± 0.012	$\mathbf{0.086} \pm 0.084$	$\underline{0.116} \pm 0.093$
mixedII300	HV	0.673 ± 0.070	0.564 ± 0.002	0.551 ± 0.004	$\mathbf{0.872} \pm 0.048$	$\underline{0.844} \pm 0.066$
	IGD	0.270 ± 0.088	0.450 ± 0.003	0.454 ± 0.004	$\mathbf{0.083} \pm 0.084$	$\underline{0.110} \pm 0.093$
randomAB300	HV	0.707 ± 0.057	0.571 ± 0.005	0.576 ± 0.005	$\mathbf{0.899} \pm 0.037$	$\underline{0.868} \pm 0.060$
	IGD	0.221 ± 0.059	0.475 ± 0.018	0.470 ± 0.015	$\mathbf{0.081} \pm 0.061$	$\underline{0.106} \pm 0.055$
randomCD300	HV	0.701 ± 0.064	0.575 ± 0.003	0.584 ± 0.001	$\mathbf{0.900} \pm 0.036$	$\underline{0.868} \pm 0.060$
	IGD	0.227 ± 0.054	0.472 ± 0.009	0.471 ± 0.012	$\mathbf{0.084} \pm 0.063$	$\underline{0.110} \pm 0.057$
randomEF300	HV	0.704 ± 0.055	0.580 ± 0.002	0.591 ± 0.001	$\mathbf{0.900} \pm 0.039$	$\underline{0.871} \pm 0.060$
	IGD	0.230 ± 0.068	0.456 ± 0.002	0.461 ± 0.001	$\mathbf{0.084} \pm 0.072$	$\underline{0.111} \pm 0.063$
euclidAB500	HV	0.726 ± 0.061	0.553 ± 0.003	0.564 ± 0.004	$\mathbf{0.873} \pm 0.052$	$\underline{0.850} \pm 0.066$
	IGD	0.261 ± 0.092	0.459 ± 0.010	0.454 ± 0.006	$\mathbf{0.070} \pm 0.080$	$\underline{0.098} \pm 0.087$
euclidCD500	HV	0.722 ± 0.074	0.550 ± 0.003	0.550 ± 0.012	$\mathbf{0.870} \pm 0.050$	$\underline{0.845} \pm 0.069$
	IGD	0.255 ± 0.082	0.464 ± 0.006	0.454 ± 0.003	$\mathbf{0.067} \pm 0.076$	$\underline{0.095} \pm 0.086$
euclidEF500	HV	0.718 ± 0.064	0.544 ± 0.002	0.549 ± 0.001	$\mathbf{0.866} \pm 0.050$	$\underline{0.841} \pm 0.069$
	IGD	0.258 ± 0.095	0.480 ± 0.005	0.473 ± 0.010	$\mathbf{0.068} \pm 0.074$	$\underline{0.099} \pm 0.090$
mixedGG500	HV	0.706 ± 0.054	0.556 ± 0.003	0.575 ± 0.001	$\mathbf{0.892} \pm 0.047$	$\underline{0.867} \pm 0.057$
	IGD	0.268 ± 0.080	0.487 ± 0.007	0.478 ± 0.010	$\mathbf{0.085} \pm 0.083$	$\underline{0.108} \pm 0.074$
mixedHH500	HV	0.703 ± 0.065	0.550 ± 0.003	0.538 ± 0.003	$\mathbf{0.892} \pm 0.047$	$\underline{0.866} \pm 0.061$
	IGD	0.282 ± 0.091	0.491 ± 0.006	0.479 ± 0.005	$\mathbf{0.089} \pm 0.094$	$\underline{0.116} \pm 0.086$
mixedII500	HV	0.707 ± 0.060	0.544 ± 0.004	0.559 ± 0.001	$\mathbf{0.894} \pm 0.046$	$\underline{0.869} \pm 0.057$
	IGD	0.273 ± 0.068	0.517 ± 0.012	0.517 ± 0.014	$\mathbf{0.088} \pm 0.089$	$\underline{0.112} \pm 0.072$
randomAB500	HV	0.716 ± 0.060	0.539 ± 0.001	0.553 ± 0.001	$\mathbf{0.913} \pm 0.040$	$\underline{0.887} \pm 0.057$
	IGD	0.223 ± 0.056	0.575 ± 0.008	0.569 ± 0.012	$\mathbf{0.083} \pm 0.069$	$\underline{0.114} \pm 0.052$
randomCD500	HV	0.718 ± 0.052	0.544 ± 0.002	0.553 ± 0.001	$\mathbf{0.915} \pm 0.037$	$\underline{0.889} \pm 0.056$
	IGD	0.224 ± 0.054	0.564 ± 0.009	0.563 ± 0.012	$\mathbf{0.086} \pm 0.067$	$\underline{0.113} \pm 0.050$
randomEF500	HV	0.735 ± 0.058	0.558 ± 0.002	0.576 ± 0.001	$\mathbf{0.919} \pm 0.034$	$\underline{0.893} \pm 0.055$
	IGD	0.208 ± 0.051	0.532 ± 0.007	0.538 ± 0.006	$\mathbf{0.079} \pm 0.058$	$\underline{0.106} \pm 0.054$

Table 10: Per-instance performance comparison between MOSAIC and LLM-driven multi-operator design methods on the Bi-KP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. The best and second-best mean values are highlighted in bold and underlined, respectively.

Instance	Metric	G-LNS	$A_{\text{best}}^{\text{E2OC}}$+NSGA-II	$A_{\text{best}}^{\text{E2OC}}$+MOEA/D	$A_{\text{best}}^{\text{MOSAIC}}$+NSGA-II	$A_{\text{best}}^{\text{MOSAIC}}$+MOEA/D
2KP200-1A	HV	0.518 ± 0.149	0.499 ± 0.115	0.506 ± 0.116	$\mathbf{0.595} \pm 0.015$	$\underline{0.589} \pm 0.014$
	IGD	0.283 ± 0.148	0.259 ± 0.061	0.265 ± 0.061	$\mathbf{0.199} \pm 0.013$	$\underline{0.206} \pm 0.011$
2KP200-1B	HV	0.566 ± 0.181	0.531 ± 0.129	0.528 ± 0.125	$\mathbf{0.658} \pm 0.009$	$\underline{0.653} \pm 0.011$
	IGD	0.258 ± 0.139	0.263 ± 0.058	0.260 ± 0.061	$\mathbf{0.182} \pm 0.021$	$\underline{0.196} \pm 0.017$
2KP200-1C	HV	0.516 ± 0.146	0.565 ± 0.100	$\underline{0.574} \pm 0.097$	$\mathbf{0.593} \pm 0.016$	0.591 ± 0.013
	IGD	0.272 ± 0.135	$\mathbf{0.179} \pm 0.063$	$\underline{0.187} \pm 0.065$	0.197 ± 0.020	0.192 ± 0.023
2KP200-1D	HV	0.535 ± 0.181	0.568 ± 0.074	$\underline{0.576} \pm 0.071$	$\mathbf{0.619} \pm 0.006$	0.615 ± 0.010
	IGD	0.281 ± 0.163	$\mathbf{0.190} \pm 0.057$	$\underline{0.197} \pm 0.064$	0.205 ± 0.011	0.203 ± 0.021
2KP250-1A	HV	0.524 ± 0.168	0.531 ± 0.138	0.529 ± 0.133	$\mathbf{0.612} \pm 0.007$	$\underline{0.607} \pm 0.007$
	IGD	0.282 ± 0.142	0.245 ± 0.089	0.243 ± 0.094	$\underline{0.205} \pm 0.011$	$\mathbf{0.204} \pm 0.011$
2KP250-1B	HV	0.534 ± 0.132	0.506 ± 0.118	0.502 ± 0.117	$\mathbf{0.594} \pm 0.009$	$\underline{0.589} \pm 0.008$
	IGD	0.266 ± 0.100	0.255 ± 0.063	0.252 ± 0.066	$\mathbf{0.203} \pm 0.021$	$\underline{0.220} \pm 0.012$
2KP250-1C	HV	0.464 ± 0.184	0.482 ± 0.120	0.476 ± 0.113	$\mathbf{0.558} \pm 0.006$	$\underline{0.551} \pm 0.011$
	IGD	0.307 ± 0.145	0.271 ± 0.061	0.266 ± 0.068	$\mathbf{0.251} \pm 0.007$	$\underline{0.257} \pm 0.016$
2KP250-1D	HV	0.529 ± 0.185	0.547 ± 0.143	0.543 ± 0.158	$\mathbf{0.638} \pm 0.009$	$\underline{0.633} \pm 0.008$
	IGD	0.294 ± 0.123	0.230 ± 0.107	0.227 ± 0.093	$\underline{0.203} \pm 0.013$	$\mathbf{0.198} \pm 0.018$
2KP300-1A	HV	0.509 ± 0.199	0.500 ± 0.147	0.499 ± 0.154	$\underline{0.603} \pm 0.013$	$\mathbf{0.605} \pm 0.009$
	IGD	0.300 ± 0.181	0.265 ± 0.097	0.265 ± 0.094	$\mathbf{0.208} \pm 0.016$	$\mathbf{0.208} \pm 0.009$
2KP300-1B	HV	0.513 ± 0.183	0.463 ± 0.202	0.462 ± 0.204	$\underline{0.593} \pm 0.010$	$\mathbf{0.599} \pm 0.014$
	IGD	0.302 ± 0.177	0.317 ± 0.178	0.317 ± 0.174	$\underline{0.209} \pm 0.019$	$\mathbf{0.203} \pm 0.010$
2KP300-1C	HV	0.488 ± 0.208	0.510 ± 0.172	0.504 ± 0.171	$\mathbf{0.591} \pm 0.009$	$\underline{0.586} \pm 0.012$
	IGD	0.328 ± 0.199	0.261 ± 0.129	0.256 ± 0.131	$\mathbf{0.237} \pm 0.008$	$\underline{0.244} \pm 0.014$
2KP300-1D	HV	0.503 ± 0.188	0.421 ± 0.263	0.424 ± 0.267	$\mathbf{0.594} \pm 0.013$	$\mathbf{0.594} \pm 0.011$
	IGD	0.294 ± 0.158	0.418 ± 0.364	0.420 ± 0.364	$\mathbf{0.219} \pm 0.013$	$\underline{0.226} \pm 0.011$
2KP350-1A	HV	0.534 ± 0.143	0.505 ± 0.184	0.507 ± 0.177	$\underline{0.601} \pm 0.013$	$\mathbf{0.605} \pm 0.016$
	IGD	0.280 ± 0.121	0.283 ± 0.135	0.285 ± 0.141	$\underline{0.225} \pm 0.010$	$\mathbf{0.216} \pm 0.013$
2KP350-1B	HV	0.494 ± 0.208	0.489 ± 0.195	0.490 ± 0.192	$\underline{0.593} \pm 0.006$	$\mathbf{0.597} \pm 0.009$
	IGD	0.326 ± 0.179	0.303 ± 0.154	0.303 ± 0.157	$\underline{0.228} \pm 0.009$	$\mathbf{0.225} \pm 0.013$
2KP350-1C	HV	0.507 ± 0.183	0.504 ± 0.163	0.499 ± 0.163	$\mathbf{0.596} \pm 0.012$	$\underline{0.593} \pm 0.014$
	IGD	0.308 ± 0.158	0.269 ± 0.107	0.265 ± 0.106	$\mathbf{0.223} \pm 0.010$	$\underline{0.226} \pm 0.010$
2KP350-1D	HV	0.522 ± 0.201	0.426 ± 0.267	0.419 ± 0.262	$\underline{0.605} \pm 0.014$	$\mathbf{0.607} \pm 0.016$
	IGD	0.284 ± 0.151	0.428 ± 0.378	0.423 ± 0.381	$\underline{0.223} \pm 0.015$	$\mathbf{0.219} \pm 0.011$
2KP400-1A	HV	0.507 ± 0.178	0.468 ± 0.180	0.469 ± 0.175	$\underline{0.585} \pm 0.007$	$\mathbf{0.587} \pm 0.009$
	IGD	0.293 ± 0.144	0.310 ± 0.122	0.310 ± 0.126	$\mathbf{0.229} \pm 0.009$	$\underline{0.231} \pm 0.013$
2KP400-1B	HV	0.486 ± 0.208	0.438 ± 0.228	0.437 ± 0.222	$\underline{0.596} \pm 0.008$	$\mathbf{0.597} \pm 0.015$
	IGD	0.318 ± 0.191	0.346 ± 0.192	0.346 ± 0.196	$\mathbf{0.207} \pm 0.010$	$\underline{0.212} \pm 0.007$
2KP400-1C	HV	0.484 ± 0.210	0.458 ± 0.200	0.457 ± 0.212	$\underline{0.585} \pm 0.007$	$\mathbf{0.587} \pm 0.006$
	IGD	0.330 ± 0.179	0.310 ± 0.146	0.309 ± 0.137	$\underline{0.241} \pm 0.012$	$\mathbf{0.232} \pm 0.010$
2KP400-1D	HV	0.480 ± 0.215	0.533 ± 0.134	0.528 ± 0.125	$\mathbf{0.595} \pm 0.004$	$\underline{0.589} \pm 0.010$
	IGD	0.332 ± 0.195	0.239 ± 0.084	0.235 ± 0.094	$\mathbf{0.221} \pm 0.009$	$\underline{0.225} \pm 0.018$
2KP450-1A	HV	0.489 ± 0.182	0.463 ± 0.200	0.467 ± 0.197	$\mathbf{0.572} \pm 0.004$	$\mathbf{0.572} \pm 0.012$
	IGD	0.300 ± 0.166	0.310 ± 0.182	0.314 ± 0.185	$\mathbf{0.216} \pm 0.012$	$\underline{0.221} \pm 0.015$
2KP450-1B	HV	0.455 ± 0.209	0.439 ± 0.191	0.444 ± 0.191	$\mathbf{0.563} \pm 0.005$	$\underline{0.561} \pm 0.006$
	IGD	0.333 ± 0.208	0.317 ± 0.169	0.322 ± 0.169	$\underline{0.217} \pm 0.009$	$\mathbf{0.212} \pm 0.013$
2KP450-1C	HV	0.496 ± 0.194	0.423 ± 0.256	0.416 ± 0.252	$\underline{0.577} \pm 0.009$	$\mathbf{0.583} \pm 0.013$
	IGD	0.300 ± 0.169	0.381 ± 0.320	0.375 ± 0.324	$\mathbf{0.221} \pm 0.016$	$\underline{0.225} \pm 0.012$
2KP450-1D	HV	0.496 ± 0.145	0.416 ± 0.242	0.420 ± 0.244	$\underline{0.545} \pm 0.016$	$\mathbf{0.553} \pm 0.029$
	IGD	0.276 ± 0.138	0.480 ± 0.589	0.484 ± 0.587	$\underline{0.212} \pm 0.016$	$\mathbf{0.204} \pm 0.023$
2KP500-1A	HV	0.501 ± 0.189	0.439 ± 0.243	0.442 ± 0.247	$\underline{0.590} \pm 0.006$	$\mathbf{0.592} \pm 0.012$
	IGD	0.317 ± 0.163	0.369 ± 0.263	0.372 ± 0.259	$\underline{0.230} \pm 0.014$	$\mathbf{0.229} \pm 0.012$
2KP500-1B	HV	0.471 ± 0.182	0.372 ± 0.249	0.373 ± 0.248	$\mathbf{0.556} \pm 0.009$	$\mathbf{0.556} \pm 0.014$
	IGD	0.317 ± 0.177	0.406 ± 0.228	0.407 ± 0.229	$\underline{0.227} \pm 0.013$	$\mathbf{0.223} \pm 0.015$
2KP500-1C	HV	0.492 ± 0.162	0.365 ± 0.305	0.368 ± 0.308	$\underline{0.565} \pm 0.006$	$\mathbf{0.568} \pm 0.019$
	IGD	0.297 ± 0.153	0.463 ± 0.428	0.462 ± 0.420	$\mathbf{0.207} \pm 0.023$	$\underline{0.213} \pm 0.011$
2KP500-1D	HV	0.494 ± 0.185	0.378 ± 0.303	0.379 ± 0.306	$\mathbf{0.586} \pm 0.007$	$\underline{0.585} \pm 0.016$
	IGD	0.320 ± 0.161	0.479 ± 0.390	0.477 ± 0.383	$\mathbf{0.227} \pm 0.011$	$\underline{0.229} \pm 0.009$

Table 11: Per-instance performance comparison between MOSAIC and LLM-driven multi-operator design methods on the Bi-GVRP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. The best and second-best mean values are highlighted in bold and underlined, respectively.

Instance	Metric	G-LNS	$A_{\text{best}}^{\text{E2OC}}$+NSGA-II	$A_{\text{best}}^{\text{E2OC}}$+MOEA/D	$A_{\text{best}}^{\text{MOSAIC}}$+NSGA-II	$A_{\text{best}}^{\text{MOSAIC}}$+MOEA/D
CMT11	HV	0.311 ± 0.009	0.725 ± 0.155	0.730 ± 0.161	$\mathbf{0.816} \pm 0.021$	$\underline{0.813} \pm 0.030$
	IGD	0.533 ± 0.016	0.113 ± 0.141	0.113 ± 0.138	$\mathbf{0.033} \pm 0.017$	$\underline{0.036} \pm 0.024$
CMT12	HV	0.180 ± 0.007	0.569 ± 0.162	0.565 ± 0.160	$\mathbf{0.695} \pm 0.043$	$\underline{0.686} \pm 0.048$
	IGD	0.659 ± 0.013	0.187 ± 0.172	0.185 ± 0.168	$\mathbf{0.063} \pm 0.037$	$\underline{0.071} \pm 0.042$
CMT4	HV	0.189 ± 0.008	0.325 ± 0.034	0.315 ± 0.027	$\mathbf{0.579} \pm 0.199$	$\underline{0.510} \pm 0.200$
	IGD	0.644 ± 0.016	0.439 ± 0.042	0.437 ± 0.048	$\mathbf{0.182} \pm 0.194$	$\underline{0.250} \pm 0.195$
CMT5	HV	0.178 ± 0.001	0.298 ± 0.040	0.301 ± 0.040	$\mathbf{0.573} \pm 0.220$	$\underline{0.503} \pm 0.219$
	IGD	0.674 ± 0.003	0.477 ± 0.052	0.474 ± 0.055	$\mathbf{0.195} \pm 0.217$	$\underline{0.265} \pm 0.215$
Golden_10	HV	0.466 ± 0.007	0.733 ± 0.126	0.725 ± 0.120	$\mathbf{0.840} \pm 0.045$	$\underline{0.825} \pm 0.045$
	IGD	0.375 ± 0.008	0.132 ± 0.113	0.131 ± 0.110	$\mathbf{0.042} \pm 0.035$	$\underline{0.054} \pm 0.036$
Golden_11	HV	0.465 ± 0.002	0.746 ± 0.140	0.767 ± 0.147	$\mathbf{0.854} \pm 0.042$	$\underline{0.839} \pm 0.042$
	IGD	0.384 ± 0.001	0.130 ± 0.125	0.134 ± 0.132	$\mathbf{0.039} \pm 0.032$	$\underline{0.050} \pm 0.033$
Golden_12	HV	0.468 ± 0.000	0.756 ± 0.148	0.758 ± 0.150	$\mathbf{0.865} \pm 0.039$	$\underline{0.853} \pm 0.038$
	IGD	0.385 ± 0.000	0.129 ± 0.132	0.132 ± 0.138	$\mathbf{0.036} \pm 0.030$	$\underline{0.046} \pm 0.030$
Golden_14	HV	0.184 ± 0.008	0.660 ± 0.246	0.664 ± 0.253	$\underline{0.764} \pm 0.014$	$\mathbf{0.769} \pm 0.007$
	IGD	0.666 ± 0.019	0.137 ± 0.259	0.134 ± 0.252	$\underline{0.027} \pm 0.012$	$\mathbf{0.023} \pm 0.005$
Golden_15	HV	0.177 ± 0.004	0.674 ± 0.264	0.674 ± 0.267	$\underline{0.786} \pm 0.012$	$\mathbf{0.789} \pm 0.006$
	IGD	0.692 ± 0.011	0.143 ± 0.280	0.142 ± 0.278	$\underline{0.023} \pm 0.010$	$\mathbf{0.021} \pm 0.005$
Golden_16	HV	0.175 ± 0.003	0.687 ± 0.271	0.691 ± 0.272	$\underline{0.802} \pm 0.012$	$\mathbf{0.806} \pm 0.006$
	IGD	0.713 ± 0.005	0.146 ± 0.288	0.143 ± 0.280	$\underline{0.022} \pm 0.010$	$\mathbf{0.019} \pm 0.004$
Golden_18	HV	0.173 ± 0.006	0.431 ± 0.123	0.434 ± 0.123	$\mathbf{0.633} \pm 0.134$	$\underline{0.586} \pm 0.136$
	IGD	0.666 ± 0.009	0.321 ± 0.148	0.317 ± 0.141	$\mathbf{0.121} \pm 0.123$	$\underline{0.164} \pm 0.124$
Golden_19	HV	0.163 ± 0.002	0.422 ± 0.136	0.421 ± 0.142	$\mathbf{0.633} \pm 0.135$	$\underline{0.588} \pm 0.136$
	IGD	0.682 ± 0.006	0.336 ± 0.174	0.344 ± 0.187	$\mathbf{0.120} \pm 0.124$	$\underline{0.162} \pm 0.124$
Golden_2	HV	0.151 ± 0.001	0.675 ± 0.272	0.670 ± 0.274	$\underline{0.760} \pm 0.088$	$\mathbf{0.800} \pm 0.006$
	IGD	0.732 ± 0.002	0.165 ± 0.286	0.172 ± 0.301	$\underline{0.070} \pm 0.078$	$\mathbf{0.034} \pm 0.005$
Golden_20	HV	0.154 ± 0.002	0.424 ± 0.134	0.420 ± 0.131	$\mathbf{0.633} \pm 0.137$	$\underline{0.588} \pm 0.138$
	IGD	0.702 ± 0.004	0.332 ± 0.168	0.337 ± 0.180	$\mathbf{0.121} \pm 0.125$	$\underline{0.162} \pm 0.126$
Golden_4	HV	0.156 ± 0.004	0.691 ± 0.285	0.686 ± 0.285	$\underline{0.799} \pm 0.079$	$\mathbf{0.837} \pm 0.025$
	IGD	0.753 ± 0.006	0.183 ± 0.298	0.176 ± 0.285	$\underline{0.067} \pm 0.068$	$\mathbf{0.036} \pm 0.020$
Golden_6	HV	0.168 ± 0.007	0.652 ± 0.246	0.647 ± 0.249	$\underline{0.767} \pm 0.065$	$\mathbf{0.792} \pm 0.042$
	IGD	0.713 ± 0.011	0.189 ± 0.258	0.190 ± 0.259	$\underline{0.072} \pm 0.056$	$\mathbf{0.050} \pm 0.034$
Golden_7	HV	0.161 ± 0.001	0.677 ± 0.271	0.665 ± 0.261	$\underline{0.782} \pm 0.075$	$\mathbf{0.817} \pm 0.026$
	IGD	0.731 ± 0.001	0.178 ± 0.284	0.177 ± 0.279	$\underline{0.066} \pm 0.066$	$\mathbf{0.036} \pm 0.021$
Golden_8	HV	0.155 ± 0.005	0.695 ± 0.288	0.697 ± 0.292	$\underline{0.785} \pm 0.086$	$\mathbf{0.829} \pm 0.007$
	IGD	0.745 ± 0.008	0.171 ± 0.301	0.178 ± 0.314	$\underline{0.069} \pm 0.075$	$\mathbf{0.032} \pm 0.006$

Table 12: Per-instance performance comparison between MOSAIC and LLM-driven single-operator design methods on the Bi-TSP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. The best and second-best mean values are highlighted in bold and underlined, respectively.

Instance	Metric	EoH	FunSearch	ReEvo	MCTS-AHD	$A^{\text{MOSAIC}}_{\text{best}}$+NSGA-II	$A^{\text{MOSAIC}}_{\text{best}}$+MOEA/D
kroAB100	HV	0.766 ± 0.053	0.841 ± 0.002	$\underline{0.843} \pm 0.002$	0.840 ± 0.007	$\mathbf{0.857} \pm 0.041$	0.825 ± 0.056
	IGD	0.170 ± 0.052	0.102 ± 0.001	0.100 ± 0.005	0.096 ± 0.015	$\mathbf{0.062} \pm 0.052$	$\underline{0.094} \pm 0.070$
kroAC100	HV	0.753 ± 0.063	$\underline{0.840} \pm 0.005$	$\underline{0.840} \pm 0.002$	0.835 ± 0.011	$\mathbf{0.852} \pm 0.044$	0.821 ± 0.056
	IGD	0.155 ± 0.053	0.083 ± 0.006	$\underline{0.077} \pm 0.004$	0.087 ± 0.010	$\mathbf{0.053} \pm 0.047$	0.079 ± 0.060
kroAD100	HV	0.736 ± 0.061	$\underline{0.823} \pm 0.011$	0.811 ± 0.006	0.801 ± 0.006	$\mathbf{0.837} \pm 0.046$	0.806 ± 0.057
	IGD	0.156 ± 0.063	$\underline{0.075} \pm 0.008$	0.089 ± 0.008	0.091 ± 0.006	$\mathbf{0.052} \pm 0.049$	0.079 ± 0.060
kroAE100	HV	0.765 ± 0.059	$\underline{0.835} \pm 0.003$	0.834 ± 0.002	0.826 ± 0.006	$\mathbf{0.853} \pm 0.045$	0.825 ± 0.054
	IGD	0.153 ± 0.060	0.081 ± 0.005	0.079 ± 0.004	0.093 ± 0.004	$\mathbf{0.055} \pm 0.054$	$\underline{0.077} \pm 0.061$
kroBC100	HV	0.748 ± 0.051	$\underline{0.821} \pm 0.006$	0.814 ± 0.006	0.809 ± 0.001	$\mathbf{0.842} \pm 0.048$	0.818 ± 0.049
	IGD	0.150 ± 0.054	0.079 ± 0.006	0.083 ± 0.005	0.091 ± 0.006	$\mathbf{0.054} \pm 0.054$	$\underline{0.076} \pm 0.062$
kroBD100	HV	0.741 ± 0.050	$\underline{0.823} \pm 0.002$	0.815 ± 0.003	0.803 ± 0.003	$\mathbf{0.837} \pm 0.040$	0.806 ± 0.055
	IGD	0.154 ± 0.057	$\underline{0.069} \pm 0.002$	0.074 ± 0.003	0.083 ± 0.004	$\mathbf{0.049} \pm 0.042$	0.078 ± 0.068
kroBE100	HV	0.753 ± 0.045	$\underline{0.825} \pm 0.004$	0.741 ± 0.113	0.815 ± 0.007	$\mathbf{0.846} \pm 0.044$	0.817 ± 0.049
	IGD	0.165 ± 0.057	0.083 ± 0.005	0.150 ± 0.096	0.090 ± 0.010	$\mathbf{0.054} \pm 0.049$	$\underline{0.080} \pm 0.065$
kroCD100	HV	0.753 ± 0.052	$\underline{0.835} \pm 0.003$	$\underline{0.835} \pm 0.002$	0.831 ± 0.008	$\mathbf{0.850} \pm 0.044$	0.820 ± 0.061
	IGD	0.171 ± 0.061	$\underline{0.079} \pm 0.003$	0.080 ± 0.005	0.085 ± 0.008	$\mathbf{0.056} \pm 0.050$	0.086 ± 0.072
kroCE100	HV	0.746 ± 0.043	0.833 ± 0.004	$\underline{0.835} \pm 0.003$	0.831 ± 0.010	$\mathbf{0.850} \pm 0.045$	0.823 ± 0.056
	IGD	0.177 ± 0.054	0.087 ± 0.007	0.085 ± 0.006	0.086 ± 0.014	$\mathbf{0.057} \pm 0.057$	$\underline{0.088} \pm 0.075$
kroDE100	HV	0.752 ± 0.048	0.825 ± 0.004	0.822 ± 0.006	0.822 ± 0.002	$\mathbf{0.848} \pm 0.045$	$\underline{0.822} \pm 0.046$
	IGD	0.157 ± 0.055	0.079 ± 0.001	0.081 ± 0.009	0.079 ± 0.004	$\mathbf{0.049} \pm 0.047$	$\underline{0.072} \pm 0.057$
kroAB150	HV	0.685 ± 0.052	0.770 ± 0.106	$\underline{0.841} \pm 0.006$	0.838 ± 0.004	$\mathbf{0.859} \pm 0.043$	0.830 ± 0.064
	IGD	0.240 ± 0.034	0.152 ± 0.092	0.094 ± 0.008	0.095 ± 0.008	$\mathbf{0.056} \pm 0.055$	$\underline{0.087} \pm 0.074$
kroAB200	HV	0.767 ± 0.060	$\underline{0.845} \pm 0.006$	0.841 ± 0.004	0.835 ± 0.003	$\mathbf{0.863} \pm 0.046$	0.834 ± 0.067
	IGD	0.170 ± 0.058	0.092 ± 0.005	0.095 ± 0.004	0.100 ± 0.002	$\mathbf{0.058} \pm 0.053$	$\underline{0.084} \pm 0.071$
euclidAB300	HV	0.744 ± 0.057	$\underline{0.827} \pm 0.006$	0.822 ± 0.003	0.823 ± 0.011	$\mathbf{0.851} \pm 0.052$	0.826 ± 0.067
	IGD	0.220 ± 0.074	0.115 ± 0.006	0.122 ± 0.004	0.116 ± 0.020	$\mathbf{0.067} \pm 0.079$	$\underline{0.098} \pm 0.093$
euclidCD300	HV	0.742 ± 0.051	$\underline{0.828} \pm 0.004$	0.825 ± 0.002	0.827 ± 0.007	$\mathbf{0.854} \pm 0.049$	0.827 ± 0.068
	IGD	0.199 ± 0.062	0.100 ± 0.006	0.104 ± 0.001	0.101 ± 0.011	$\mathbf{0.059} \pm 0.063$	$\underline{0.088} \pm 0.081$
euclidEF300	HV	0.744 ± 0.057	$\underline{0.827} \pm 0.008$	0.819 ± 0.007	0.817 ± 0.004	$\mathbf{0.848} \pm 0.053$	0.820 ± 0.072
	IGD	0.205 ± 0.071	0.109 ± 0.009	0.117 ± 0.007	0.117 ± 0.009	$\mathbf{0.065} \pm 0.073$	$\underline{0.093} \pm 0.091$
mixedGG300	HV	0.791 ± 0.048	0.855 ± 0.004	$\underline{0.856} \pm 0.004$	0.853 ± 0.004	$\mathbf{0.882} \pm 0.042$	0.851 ± 0.063
	IGD	0.231 ± 0.055	0.160 ± 0.006	0.160 ± 0.006	0.161 ± 0.006	$\mathbf{0.082} \pm 0.078$	$\underline{0.109} \pm 0.082$
mixedHH300	HV	0.783 ± 0.049	$\underline{0.852} \pm 0.005$	$\underline{0.852} \pm 0.003$	0.850 ± 0.002	$\mathbf{0.876} \pm 0.047$	0.850 ± 0.060
	IGD	0.239 ± 0.054	0.163 ± 0.003	0.163 ± 0.003	0.164 ± 0.005	$\mathbf{0.086} \pm 0.084$	$\underline{0.116} \pm 0.093$
mixedII300	HV	0.783 ± 0.052	0.846 ± 0.003	$\underline{0.848} \pm 0.003$	0.845 ± 0.004	$\mathbf{0.872} \pm 0.048$	0.844 ± 0.066
	IGD	0.228 ± 0.061	0.153 ± 0.002	0.152 ± 0.006	0.152 ± 0.005	$\mathbf{0.083} \pm 0.084$	$\underline{0.110} \pm 0.093$
randomAB300	HV	0.844 ± 0.038	$\underline{0.897} \pm 0.002$	0.895 ± 0.002	0.894 ± 0.008	$\mathbf{0.899} \pm 0.037$	0.868 ± 0.060
	IGD	0.181 ± 0.045	0.125 ± 0.005	0.127 ± 0.005	0.127 ± 0.013	$\mathbf{0.081} \pm 0.061$	$\underline{0.106} \pm 0.055$
randomCD300	HV	0.843 ± 0.037	$\mathbf{0.900} \pm 0.003$	0.898 ± 0.002	$\underline{0.899} \pm 0.008$	$\mathbf{0.900} \pm 0.036$	0.868 ± 0.060
	IGD	0.191 ± 0.041	0.133 ± 0.004	0.136 ± 0.005	0.132 ± 0.012	$\mathbf{0.084} \pm 0.063$	$\underline{0.110} \pm 0.057$
randomEF300	HV	0.851 ± 0.038	$\mathbf{0.901} \pm 0.002$	0.899 ± 0.001	0.897 ± 0.005	$\underline{0.900} \pm 0.039$	0.871 ± 0.060
	IGD	0.198 ± 0.046	0.136 ± 0.005	0.140 ± 0.001	0.135 ± 0.010	$\mathbf{0.084} \pm 0.072$	$\underline{0.111} \pm 0.063$
euclidAB500	HV	0.773 ± 0.053	0.849 ± 0.002	0.843 ± 0.002	0.843 ± 0.002	$\mathbf{0.873} \pm 0.052$	$\underline{0.850} \pm 0.066$
	IGD	0.213 ± 0.062	0.128 ± 0.007	0.134 ± 0.004	0.129 ± 0.005	$\mathbf{0.070} \pm 0.080$	$\underline{0.098} \pm 0.087$
euclidCD500	HV	0.769 ± 0.061	0.840 ± 0.004	0.843 ± 0.003	0.841 ± 0.007	$\mathbf{0.870} \pm 0.050$	$\underline{0.845} \pm 0.069$
	IGD	0.206 ± 0.071	0.128 ± 0.009	0.126 ± 0.001	0.121 ± 0.014	$\mathbf{0.067} \pm 0.076$	$\underline{0.095} \pm 0.086$
euclidEF500	HV	0.761 ± 0.056	$\underline{0.841} \pm 0.005$	0.831 ± 0.007	0.823 ± 0.003	$\mathbf{0.866} \pm 0.050$	$\underline{0.841} \pm 0.069$
	IGD	0.216 ± 0.066	0.128 ± 0.007	0.135 ± 0.006	0.141 ± 0.009	$\mathbf{0.068} \pm 0.074$	$\underline{0.099} \pm 0.090$
mixedGG500	HV	0.805 ± 0.047	0.868 ± 0.003	0.869 ± 0.002	$\underline{0.870} \pm 0.002$	$\mathbf{0.892} \pm 0.047$	0.867 ± 0.057
	IGD	0.230 ± 0.052	0.163 ± 0.002	0.162 ± 0.003	0.157 ± 0.003	$\mathbf{0.085} \pm 0.083$	$\underline{0.108} \pm 0.074$
mixedHH500	HV	0.808 ± 0.047	$\underline{0.872} \pm 0.003$	0.868 ± 0.002	0.867 ± 0.001	$\mathbf{0.892} \pm 0.047$	0.866 ± 0.061
	IGD	0.244 ± 0.053	0.171 ± 0.003	0.171 ± 0.003	0.175 ± 0.003	$\mathbf{0.089} \pm 0.094$	$\underline{0.116} \pm 0.086$
mixedII500	HV	0.813 ± 0.044	$\underline{0.876} \pm 0.001$	0.871 ± 0.002	0.871 ± 0.003	$\mathbf{0.894} \pm 0.046$	0.869 ± 0.057
	IGD	0.233 ± 0.049	0.162 ± 0.004	0.169 ± 0.002	0.166 ± 0.007	$\mathbf{0.088} \pm 0.089$	$\underline{0.112} \pm 0.072$
randomAB500	HV	0.867 ± 0.036	$\mathbf{0.914} \pm 0.001$	0.910 ± 0.001	0.912 ± 0.006	$\underline{0.913} \pm 0.040$	0.887 ± 0.057
	IGD	0.188 ± 0.040	0.137 ± 0.003	0.140 ± 0.001	0.137 ± 0.009	$\mathbf{0.083} \pm 0.069$	$\underline{0.114} \pm 0.052$
randomCD500	HV	0.869 ± 0.035	$\mathbf{0.917} \pm 0.002$	$\underline{0.915} \pm 0.001$	0.912 ± 0.003	$\underline{0.915} \pm 0.037$	0.889 ± 0.056
	IGD	0.190 ± 0.037	0.141 ± 0.001	0.142 ± 0.002	0.146 ± 0.003	$\mathbf{0.086} \pm 0.067$	$\underline{0.113} \pm 0.050$
randomEF500	HV	0.873 ± 0.031	$\underline{0.919} \pm 0.003$	0.917 ± 0.003	$\mathbf{0.920} \pm 0.006$	$\underline{0.919} \pm 0.034$	0.893 ± 0.055
	IGD	0.178 ± 0.032	0.129 ± 0.006	0.130 ± 0.005	0.126 ± 0.012	$\mathbf{0.079} \pm 0.058$	$\underline{0.106} \pm 0.054$

Table 13: Per-instance performance comparison between MOSAIC and LLM-driven single-operator design methods on the Bi-KP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. The best and second-best mean values are highlighted in bold and underlined, respectively.

Instance	Metric	EoH	FunSearch	ReEvo	MCTS-AHD	$A_{\text{best}}^{\text{MOSAIC}}$+NSGA-II	$A_{\text{best}}^{\text{MOSAIC}}$+MOEA/D
2KP200-1A	HV	0.574 ± 0.004	0.572 ± 0.007	0.565 ± 0.009	0.567 ± 0.004	$\mathbf{0.595} \pm 0.015$	$\underline{0.589} \pm 0.014$
	IGD	0.207 ± 0.008	0.210 ± 0.005	0.218 ± 0.034	0.223 ± 0.015	$\mathbf{0.199} \pm 0.013$	$\underline{0.206} \pm 0.011$
2KP200-1B	HV	0.629 ± 0.010	0.634 ± 0.002	0.625 ± 0.003	0.636 ± 0.000	$\mathbf{0.658} \pm 0.009$	$\underline{0.653} \pm 0.011$
	IGD	0.227 ± 0.009	0.206 ± 0.012	0.220 ± 0.011	0.201 ± 0.014	$\mathbf{0.182} \pm 0.021$	$\underline{0.196} \pm 0.017$
2KP200-1C	HV	0.582 ± 0.013	0.581 ± 0.002	$\mathbf{0.594} \pm 0.023$	0.577 ± 0.010	$\underline{0.593} \pm 0.016$	0.591 ± 0.013
	IGD	0.211 ± 0.021	0.206 ± 0.004	$\mathbf{0.192} \pm 0.030$	0.202 ± 0.020	$\underline{0.197} \pm 0.020$	$\mathbf{0.192} \pm 0.023$
2KP200-1D	HV	0.595 ± 0.004	0.597 ± 0.005	0.598 ± 0.008	0.594 ± 0.010	$\mathbf{0.619} \pm 0.006$	$\underline{0.615} \pm 0.010$
	IGD	0.224 ± 0.009	0.222 ± 0.001	0.221 ± 0.015	0.217 ± 0.006	$\underline{0.205} \pm 0.011$	$\mathbf{0.203} \pm 0.021$
2KP250-1A	HV	0.588 ± 0.006	0.592 ± 0.003	0.579 ± 0.004	0.586 ± 0.001	$\mathbf{0.612} \pm 0.007$	$\underline{0.607} \pm 0.007$
	IGD	0.222 ± 0.014	0.229 ± 0.004	0.245 ± 0.002	0.227 ± 0.012	$\underline{0.205} \pm 0.011$	$\mathbf{0.204} \pm 0.011$
2KP250-1B	HV	0.564 ± 0.007	0.576 ± 0.005	0.581 ± 0.005	0.583 ± 0.022	$\mathbf{0.594} \pm 0.009$	$\underline{0.589} \pm 0.008$
	IGD	0.257 ± 0.010	0.230 ± 0.009	0.229 ± 0.006	0.224 ± 0.023	$\mathbf{0.203} \pm 0.021$	$\underline{0.220} \pm 0.012$
2KP250-1C	HV	0.526 ± 0.004	0.538 ± 0.002	0.524 ± 0.010	0.535 ± 0.011	$\mathbf{0.558} \pm 0.006$	$\underline{0.551} \pm 0.011$
	IGD	0.268 ± 0.001	0.259 ± 0.007	0.284 ± 0.016	0.264 ± 0.029	$\mathbf{0.251} \pm 0.007$	$\underline{0.257} \pm 0.016$
2KP250-1D	HV	0.610 ± 0.012	0.625 ± 0.001	0.615 ± 0.017	0.624 ± 0.008	$\mathbf{0.638} \pm 0.009$	$\underline{0.633} \pm 0.008$
	IGD	0.236 ± 0.023	0.209 ± 0.012	0.236 ± 0.023	$\mathbf{0.195} \pm 0.004$	0.203 ± 0.013	$\underline{0.198} \pm 0.018$
2KP300-1A	HV	0.584 ± 0.000	0.588 ± 0.004	0.584 ± 0.000	0.594 ± 0.002	$\underline{0.603} \pm 0.013$	$\mathbf{0.605} \pm 0.009$
	IGD	0.231 ± 0.007	$\underline{0.221} \pm 0.006$	0.235 ± 0.001	0.226 ± 0.002	$\mathbf{0.208} \pm 0.016$	$\mathbf{0.208} \pm 0.009$
2KP300-1B	HV	0.574 ± 0.003	0.592 ± 0.002	0.585 ± 0.002	0.577 ± 0.003	$\underline{0.593} \pm 0.010$	$\mathbf{0.599} \pm 0.014$
	IGD	0.243 ± 0.003	0.215 ± 0.002	0.215 ± 0.000	0.241 ± 0.001	$\underline{0.209} \pm 0.019$	$\mathbf{0.203} \pm 0.010$
2KP300-1C	HV	0.565 ± 0.003	0.568 ± 0.001	0.576 ± 0.005	0.578 ± 0.009	$\mathbf{0.591} \pm 0.009$	$\underline{0.586} \pm 0.012$
	IGD	0.268 ± 0.001	0.250 ± 0.016	0.250 ± 0.001	$\mathbf{0.237} \pm 0.017$	$\mathbf{0.237} \pm 0.008$	$\underline{0.244} \pm 0.014$
2KP300-1D	HV	0.571 ± 0.013	$\mathbf{0.597} \pm 0.000$	0.587 ± 0.001	0.585 ± 0.003	$\underline{0.594} \pm 0.013$	$\underline{0.594} \pm 0.011$
	IGD	0.252 ± 0.016	0.225 ± 0.007	0.236 ± 0.009	$\mathbf{0.213} \pm 0.001$	$\underline{0.219} \pm 0.013$	0.226 ± 0.011
2KP350-1A	HV	0.582 ± 0.003	0.580 ± 0.002	0.585 ± 0.007	0.587 ± 0.000	$\underline{0.601} \pm 0.013$	$\mathbf{0.605} \pm 0.016$
	IGD	0.248 ± 0.001	0.254 ± 0.002	0.242 ± 0.007	0.230 ± 0.011	$\underline{0.225} \pm 0.010$	$\mathbf{0.216} \pm 0.013$
2KP350-1B	HV	0.578 ± 0.004	0.578 ± 0.005	0.579 ± 0.004	0.581 ± 0.002	$\underline{0.593} \pm 0.006$	$\mathbf{0.597} \pm 0.009$
	IGD	0.243 ± 0.002	0.251 ± 0.004	0.256 ± 0.003	0.249 ± 0.001	$\underline{0.228} \pm 0.009$	$\mathbf{0.225} \pm 0.013$
2KP350-1C	HV	0.571 ± 0.001	0.587 ± 0.011	0.576 ± 0.000	0.577 ± 0.008	$\mathbf{0.596} \pm 0.012$	$\underline{0.593} \pm 0.014$
	IGD	0.260 ± 0.008	0.243 ± 0.017	0.253 ± 0.009	0.260 ± 0.005	$\mathbf{0.223} \pm 0.010$	$\underline{0.226} \pm 0.010$
2KP350-1D	HV	0.603 ± 0.002	0.599 ± 0.004	0.598 ± 0.012	$\mathbf{0.609} \pm 0.003$	0.605 ± 0.014	$\underline{0.607} \pm 0.016$
	IGD	0.233 ± 0.004	$\mathbf{0.214} \pm 0.005$	0.230 ± 0.010	0.235 ± 0.001	0.223 ± 0.015	$\underline{0.219} \pm 0.011$
2KP400-1A	HV	0.558 ± 0.005	0.572 ± 0.005	0.570 ± 0.001	0.574 ± 0.000	$\underline{0.585} \pm 0.007$	$\mathbf{0.587} \pm 0.009$
	IGD	0.270 ± 0.006	0.248 ± 0.003	0.248 ± 0.010	0.243 ± 0.007	$\mathbf{0.229} \pm 0.009$	$\underline{0.231} \pm 0.013$
2KP400-1B	HV	0.573 ± 0.008	0.592 ± 0.003	0.586 ± 0.001	0.591 ± 0.002	$\underline{0.596} \pm 0.008$	$\mathbf{0.597} \pm 0.015$
	IGD	0.253 ± 0.011	0.220 ± 0.007	0.234 ± 0.000	0.226 ± 0.008	$\mathbf{0.207} \pm 0.010$	$\underline{0.212} \pm 0.007$
2KP400-1C	HV	0.565 ± 0.008	0.574 ± 0.000	0.572 ± 0.004	0.571 ± 0.003	$\underline{0.585} \pm 0.007$	$\mathbf{0.587} \pm 0.006$
	IGD	0.268 ± 0.003	0.253 ± 0.001	0.263 ± 0.003	0.246 ± 0.012	$\underline{0.241} \pm 0.012$	$\mathbf{0.232} \pm 0.010$
2KP400-1D	HV	0.569 ± 0.001	0.574 ± 0.002	0.566 ± 0.004	0.578 ± 0.008	$\mathbf{0.595} \pm 0.004$	$\underline{0.589} \pm 0.010$
	IGD	0.258 ± 0.004	0.251 ± 0.003	0.260 ± 0.002	0.240 ± 0.018	$\mathbf{0.221} \pm 0.009$	$\underline{0.225} \pm 0.018$
2KP450-1A	HV	0.555 ± 0.008	$\underline{0.563} \pm 0.003$	0.558 ± 0.003	0.554 ± 0.001	$\mathbf{0.572} \pm 0.004$	$\mathbf{0.572} \pm 0.012$
	IGD	0.251 ± 0.012	0.234 ± 0.010	0.234 ± 0.001	0.237 ± 0.002	$\mathbf{0.216} \pm 0.012$	$\underline{0.221} \pm 0.015$
2KP450-1B	HV	0.545 ± 0.003	0.547 ± 0.003	0.544 ± 0.002	0.544 ± 0.000	$\mathbf{0.563} \pm 0.005$	$\underline{0.561} \pm 0.006$
	IGD	0.237 ± 0.007	0.237 ± 0.000	0.243 ± 0.003	0.237 ± 0.008	$\underline{0.217} \pm 0.009$	$\mathbf{0.212} \pm 0.013$
2KP450-1C	HV	0.563 ± 0.012	0.565 ± 0.001	0.565 ± 0.004	0.564 ± 0.001	$\underline{0.577} \pm 0.009$	$\mathbf{0.583} \pm 0.013$
	IGD	0.252 ± 0.007	0.249 ± 0.003	0.252 ± 0.007	0.255 ± 0.005	$\mathbf{0.221} \pm 0.016$	$\underline{0.225} \pm 0.012$
2KP450-1D	HV	0.539 ± 0.017	0.543 ± 0.000	0.538 ± 0.008	$\underline{0.546} \pm 0.008$	0.545 ± 0.016	$\mathbf{0.553} \pm 0.029$
	IGD	0.233 ± 0.005	0.229 ± 0.003	0.222 ± 0.002	0.216 ± 0.017	$\underline{0.212} \pm 0.016$	$\mathbf{0.204} \pm 0.023$
2KP500-1A	HV	0.575 ± 0.004	0.581 ± 0.005	0.585 ± 0.004	0.574 ± 0.000	$\underline{0.590} \pm 0.006$	$\mathbf{0.592} \pm 0.012$
	IGD	0.254 ± 0.003	0.241 ± 0.004	0.245 ± 0.005	0.262 ± 0.000	$\underline{0.230} \pm 0.014$	$\mathbf{0.229} \pm 0.012$
2KP500-1B	HV	0.539 ± 0.002	$\underline{0.550} \pm 0.000$	0.536 ± 0.002	0.543 ± 0.009	$\mathbf{0.556} \pm 0.009$	$\mathbf{0.556} \pm 0.014$
	IGD	0.254 ± 0.006	0.242 ± 0.000	0.257 ± 0.001	0.245 ± 0.023	$\underline{0.227} \pm 0.013$	$\mathbf{0.223} \pm 0.015$
2KP500-1C	HV	0.545 ± 0.003	0.554 ± 0.004	0.553 ± 0.000	0.557 ± 0.008	$\underline{0.565} \pm 0.006$	$\mathbf{0.568} \pm 0.019$
	IGD	0.254 ± 0.008	0.228 ± 0.019	0.228 ± 0.015	0.236 ± 0.013	$\mathbf{0.207} \pm 0.023$	$\underline{0.213} \pm 0.011$
2KP500-1D	HV	0.573 ± 0.006	0.577 ± 0.005	0.572 ± 0.013	0.574 ± 0.001	$\mathbf{0.586} \pm 0.007$	$\underline{0.585} \pm 0.016$
	IGD	0.247 ± 0.008	0.242 ± 0.005	0.255 ± 0.003	0.250 ± 0.005	$\mathbf{0.227} \pm 0.011$	$\underline{0.229} \pm 0.009$

Table 14: Per-instance performance comparison between MOSAIC and LLM-driven single-operator design methods on the Bi-GVRP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. The best and second-best mean values are highlighted in bold and underlined, respectively.

Instance	Metric	EoH	FunSearch	ReEvo	MCTS-AHD	$A_{\text{best}}^{\text{MOSAIC}}$+NSGA-II	$A_{\text{best}}^{\text{MOSAIC}}$+MOEA/D
CMT11	HV	0.800 ± 0.009	0.801 ± 0.005	0.801 ± 0.006	0.801 ± 0.003	$\mathbf{0.816} \pm 0.021$	$\underline{0.813} \pm 0.030$
	IGD	0.046 ± 0.008	0.045 ± 0.004	0.045 ± 0.004	0.045 ± 0.003	$\mathbf{0.033} \pm 0.017$	$\underline{0.036} \pm 0.024$
CMT12	HV	0.662 ± 0.012	0.653 ± 0.008	0.651 ± 0.004	0.650 ± 0.003	$\mathbf{0.695} \pm 0.043$	$\underline{0.686} \pm 0.048$
	IGD	0.091 ± 0.011	0.100 ± 0.008	0.101 ± 0.003	0.103 ± 0.003	$\mathbf{0.063} \pm 0.037$	$\underline{0.071} \pm 0.042$
CMT4	HV	0.366 ± 0.020	0.356 ± 0.006	0.348 ± 0.004	0.358 ± 0.007	$\mathbf{0.579} \pm 0.199$	$\underline{0.510} \pm 0.200$
	IGD	0.392 ± 0.025	0.403 ± 0.008	0.411 ± 0.004	0.399 ± 0.007	$\mathbf{0.182} \pm 0.194$	$\underline{0.250} \pm 0.195$
CMT5	HV	0.340 ± 0.008	0.327 ± 0.005	0.330 ± 0.002	0.333 ± 0.001	$\mathbf{0.573} \pm 0.220$	$\underline{0.503} \pm 0.219$
	IGD	0.425 ± 0.011	0.441 ± 0.009	0.437 ± 0.002	0.433 ± 0.002	$\mathbf{0.195} \pm 0.217$	$\underline{0.265} \pm 0.215$
Golden_10	HV	0.794 ± 0.001	0.793 ± 0.002	0.792 ± 0.002	0.792 ± 0.001	$\mathbf{0.840} \pm 0.045$	$\underline{0.825} \pm 0.045$
	IGD	0.078 ± 0.001	0.079 ± 0.002	0.079 ± 0.002	0.080 ± 0.001	$\mathbf{0.042} \pm 0.035$	$\underline{0.054} \pm 0.036$
Golden_11	HV	0.811 ± 0.001	0.809 ± 0.002	0.811 ± 0.002	0.812 ± 0.003	$\mathbf{0.854} \pm 0.042$	$\underline{0.839} \pm 0.042$
	IGD	0.072 ± 0.001	0.073 ± 0.002	0.072 ± 0.002	0.071 ± 0.002	$\mathbf{0.039} \pm 0.032$	$\underline{0.050} \pm 0.033$
Golden_12	HV	0.828 ± 0.004	0.826 ± 0.001	0.824 ± 0.001	0.825 ± 0.001	$\mathbf{0.865} \pm 0.039$	$\underline{0.853} \pm 0.038$
	IGD	0.065 ± 0.003	0.067 ± 0.001	0.068 ± 0.001	0.067 ± 0.001	$\mathbf{0.036} \pm 0.030$	$\underline{0.046} \pm 0.030$
Golden_14	HV	$\mathbf{0.774} \pm 0.000$	0.771 ± 0.002	0.771 ± 0.002	$\underline{0.772} \pm 0.002$	0.764 ± 0.014	0.769 ± 0.007
	IGD	$\mathbf{0.019} \pm 0.000$	0.021 ± 0.002	0.021 ± 0.001	$\underline{0.020} \pm 0.001$	0.027 ± 0.012	0.023 ± 0.005
Golden_15	HV	$\mathbf{0.792} \pm 0.000$	$\mathbf{0.792} \pm 0.001$	$\mathbf{0.792} \pm 0.000$	$\mathbf{0.792} \pm 0.001$	0.786 ± 0.012	$\underline{0.789} \pm 0.006$
	IGD	$\mathbf{0.018} \pm 0.000$	$\mathbf{0.018} \pm 0.000$	$\mathbf{0.018} \pm 0.000$	$\mathbf{0.018} \pm 0.001$	0.023 ± 0.010	$\underline{0.021} \pm 0.005$
Golden_16	HV	$\mathbf{0.810} \pm 0.000$	$\mathbf{0.810} \pm 0.001$	$\underline{0.809} \pm 0.001$	$\underline{0.809} \pm 0.001$	0.802 ± 0.012	0.806 ± 0.006
	IGD	$\mathbf{0.016} \pm 0.000$	$\mathbf{0.016} \pm 0.001$	$\underline{0.017} \pm 0.001$	$\underline{0.017} \pm 0.000$	0.022 ± 0.010	0.019 ± 0.004
Golden_18	HV	0.490 ± 0.003	0.489 ± 0.004	0.489 ± 0.002	0.489 ± 0.001	$\mathbf{0.633} \pm 0.134$	$\underline{0.586} \pm 0.136$
	IGD	0.251 ± 0.002	0.252 ± 0.004	0.252 ± 0.002	0.252 ± 0.000	$\mathbf{0.121} \pm 0.123$	$\underline{0.164} \pm 0.124$
Golden_19	HV	0.488 ± 0.001	0.488 ± 0.002	0.489 ± 0.001	0.486 ± 0.001	$\mathbf{0.633} \pm 0.135$	$\underline{0.588} \pm 0.136$
	IGD	0.253 ± 0.000	0.252 ± 0.002	0.251 ± 0.001	0.254 ± 0.001	$\mathbf{0.120} \pm 0.124$	$\underline{0.162} \pm 0.124$
Golden_2	HV	$\underline{0.797} \pm 0.000$	$\underline{0.797} \pm 0.001$	$\underline{0.797} \pm 0.000$	$\underline{0.797} \pm 0.000$	0.760 ± 0.088	$\mathbf{0.800} \pm 0.006$
	IGD	$\underline{0.037} \pm 0.000$	$\underline{0.037} \pm 0.001$	$\underline{0.037} \pm 0.000$	$\underline{0.037} \pm 0.000$	0.070 ± 0.078	$\mathbf{0.034} \pm 0.005$
Golden_20	HV	0.488 ± 0.003	0.487 ± 0.003	0.486 ± 0.002	0.488 ± 0.002	$\mathbf{0.633} \pm 0.137$	$\underline{0.588} \pm 0.138$
	IGD	0.253 ± 0.003	0.254 ± 0.004	0.256 ± 0.002	0.253 ± 0.001	$\mathbf{0.121} \pm 0.125$	$\underline{0.162} \pm 0.126$
Golden_4	HV	$\underline{0.820} \pm 0.000$	$\underline{0.820} \pm 0.000$	$\underline{0.820} \pm 0.000$	$\underline{0.820} \pm 0.000$	0.799 ± 0.079	$\mathbf{0.837} \pm 0.025$
	IGD	$\underline{0.049} \pm 0.000$	$\underline{0.049} \pm 0.000$	$\underline{0.049} \pm 0.000$	$\underline{0.049} \pm 0.000$	0.067 ± 0.068	$\mathbf{0.036} \pm 0.020$
Golden_6	HV	0.763 ± 0.000	0.763 ± 0.000	0.763 ± 0.000	0.763 ± 0.000	$\underline{0.767} \pm 0.065$	$\mathbf{0.792} \pm 0.042$
	IGD	0.073 ± 0.000	0.073 ± 0.000	0.073 ± 0.000	0.073 ± 0.000	$\underline{0.072} \pm 0.056$	$\mathbf{0.050} \pm 0.034$
Golden_7	HV	$\underline{0.799} \pm 0.000$	$\underline{0.799} \pm 0.000$	$\underline{0.799} \pm 0.000$	$\underline{0.799} \pm 0.001$	0.782 ± 0.075	$\mathbf{0.817} \pm 0.026$
	IGD	$\underline{0.050} \pm 0.000$	$\underline{0.050} \pm 0.000$	$\underline{0.050} \pm 0.000$	$\underline{0.050} \pm 0.001$	0.066 ± 0.066	$\mathbf{0.036} \pm 0.021$
Golden_8	HV	0.824 ± 0.000	$\underline{0.825} \pm 0.000$	0.824 ± 0.000	$\underline{0.825} \pm 0.000$	0.785 ± 0.086	$\mathbf{0.829} \pm 0.007$
	IGD	$\underline{0.035} \pm 0.000$	$\underline{0.035} \pm 0.000$	$\underline{0.035} \pm 0.000$	$\underline{0.035} \pm 0.000$	0.069 ± 0.075	$\mathbf{0.032} \pm 0.006$

C.5 Statistical Significance Results

We provide detailed statistical significance results for the comparisons between MOSAIC and the multi-operator and single-operator design methods. Within each problem size group, two-sided Wilcoxon signed-rank tests are performed on the per-instance mean results. MOSAIC is compared with each baseline, and Holm correction is applied across baseline comparisons, separately for HV and IGD. Problem size groups containing only a single test instance are excluded from statistical testing. Table 15 reports the adjusted p-values.

Table 15: Statistical significance comparisons between MOSAIC and the baseline methods on Bi-TSP, Bi-KP, and Bi-GVRP. Each entry reports the Holm-adjusted p-value from a two-sided Wilcoxon signed-rank test on per-instance mean results. Statistically significant differences at $\alpha = 0.05$ are highlighted in bold. "–" denotes size groups excluded from statistical testing because they contain only a single test instance.

Problem	Scale	EoH	FunSearch	ReEvo	MCTS-AHD	G-LNS	E2OC
				HV			
Bi-TSP	100	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$
	150	–	–	–	–	–	–
	200	–	–	–	–	–	–
	300	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$
	500	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{3.91 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$
Bi-KP	200–300	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$
	350–400	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$
	450–500	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$
Bi-GVRP	100–199	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}
	200–483	1.00×10^{0}	1.00×10^{0}	1.00×10^{0}	1.00×10^{0}	$\mathbf{5.90 \times 10^{-3}}$	$\mathbf{5.90 \times 10^{-3}}$
				IGD			
Bi-TSP	100	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-2}}$
	150	–	–	–	–	–	–
	200	–	–	–	–	–	–
	300	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$
	500	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$	$\mathbf{2.34 \times 10^{-2}}$
Bi-KP	200–300	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{9.80 \times 10^{-3}}$	$\mathbf{2.90 \times 10^{-3}}$	$\mathbf{4.90 \times 10^{-3}}$
	350–400	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$
	450–500	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$	$\mathbf{4.69 \times 10^{-2}}$
Bi-GVRP	100–199	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}	7.50×10^{-1}
	200–483	1.00×10^{0}	1.00×10^{0}	1.00×10^{0}	1.00×10^{0}	$\mathbf{7.00 \times 10^{-4}}$	$\mathbf{7.00 \times 10^{-4}}$

C.6 Detailed Ablation Results

Tables 16–18 report detailed per-instance HV and IGD results for the ablation variants on Bi-TSP, Bi-KP, and Bi-GVRP. Results are reported as the mean and standard deviation over five independent runs. The per-instance results show consistent trends with the aggregate ablation results. Removing AR or BR generally reduces performance across the test instances, while removing both produces more pronounced degradation. Fixed-Arch also performs worse than the complete MOSAIC across the three problems. The complete MOSAIC generally achieves the strongest per-instance performance, further supporting the contributions of both module evolution and architecture evolution.

Table 16: Ablation experimental results of MOSAIC on Bi-TSP. Higher HV and lower IGD indicate better performance. The best and second-best values are highlighted in bold and underlined, respectively. Results are averaged over five independent runs.

Instance	Metric	MOSAIC w/o AR	MOSAIC w/o BR	MOSAIC w/o AR+BR	Fixed-Arch	MOSAIC
kroAB100	HV	0.836 ± 0.005	0.791 ± 0.024	0.431 ± 0.150	0.739 ± 0.019	**0.857 ± 0.041**
	IGD	0.082 ± 0.008	0.098 ± 0.007	0.372 ± 0.113	0.193 ± 0.011	**0.062 ± 0.052**
kroAC100	HV	0.831 ± 0.005	0.792 ± 0.027	0.433 ± 0.128	0.729 ± 0.016	**0.852 ± 0.044**
	IGD	0.067 ± 0.004	0.086 ± 0.009	0.337 ± 0.084	0.176 ± 0.010	**0.053 ± 0.047**
kroAD100	HV	0.817 ± 0.005	0.774 ± 0.029	0.386 ± 0.152	0.711 ± 0.008	**0.837 ± 0.046**
	IGD	0.063 ± 0.005	0.094 ± 0.011	0.392 ± 0.115	0.180 ± 0.011	**0.052 ± 0.049**
kroAE100	HV	0.835 ± 0.005	0.787 ± 0.032	0.411 ± 0.156	0.732 ± 0.007	**0.853 ± 0.045**
	IGD	0.062 ± 0.005	0.094 ± 0.010	0.377 ± 0.115	0.178 ± 0.008	**0.055 ± 0.054**
kroBC100	HV	0.826 ± 0.005	0.779 ± 0.030	0.421 ± 0.152	0.725 ± 0.013	**0.842 ± 0.048**
	IGD	0.059 ± 0.004	0.089 ± 0.013	0.357 ± 0.112	0.181 ± 0.007	**0.054 ± 0.054**
kroBD100	HV	0.818 ± 0.005	0.764 ± 0.029	0.413 ± 0.131	0.701 ± 0.021	**0.837 ± 0.040**
	IGD	0.061 ± 0.004	0.096 ± 0.014	0.357 ± 0.089	0.194 ± 0.013	**0.049 ± 0.042**
kroBE100	HV	0.827 ± 0.005	0.777 ± 0.025	0.411 ± 0.153	0.721 ± 0.011	**0.846 ± 0.044**
	IGD	0.065 ± 0.005	0.091 ± 0.009	0.376 ± 0.121	0.193 ± 0.009	**0.054 ± 0.049**
kroCD100	HV	0.831 ± 0.005	0.789 ± 0.039	0.397 ± 0.152	0.718 ± 0.022	**0.850 ± 0.044**
	IGD	0.065 ± 0.006	0.095 ± 0.016	0.403 ± 0.124	0.202 ± 0.007	**0.056 ± 0.050**
kroCE100	HV	0.829 ± 0.005	0.790 ± 0.033	0.422 ± 0.150	0.727 ± 0.020	**0.850 ± 0.045**
	IGD	0.069 ± 0.007	0.096 ± 0.012	0.380 ± 0.113	0.196 ± 0.012	**0.057 ± 0.057**
kroDE100	HV	0.827 ± 0.004	0.784 ± 0.028	0.407 ± 0.159	0.725 ± 0.018	**0.848 ± 0.045**
	IGD	0.062 ± 0.005	0.085 ± 0.014	0.368 ± 0.103	0.180 ± 0.014	**0.049 ± 0.047**
kroAB150	HV	0.832 ± 0.005	0.789 ± 0.036	0.387 ± 0.169	0.727 ± 0.010	**0.859 ± 0.043**
	IGD	0.078 ± 0.005	0.102 ± 0.010	0.412 ± 0.124	0.200 ± 0.010	**0.056 ± 0.055**
kroAB200	HV	0.836 ± 0.005	0.786 ± 0.033	0.355 ± 0.183	0.727 ± 0.016	**0.863 ± 0.046**
	IGD	0.082 ± 0.006	0.102 ± 0.011	0.442 ± 0.141	0.203 ± 0.008	**0.058 ± 0.053**
euclidAB300	HV	0.824 ± 0.005	0.761 ± 0.035	0.303 ± 0.206	0.704 ± 0.014	**0.851 ± 0.052**
	IGD	0.092 ± 0.009	0.119 ± 0.011	0.499 ± 0.136	0.260 ± 0.010	**0.067 ± 0.079**
euclidCD300	HV	0.825 ± 0.005	0.764 ± 0.038	0.320 ± 0.194	0.699 ± 0.014	**0.854 ± 0.049**
	IGD	0.083 ± 0.006	0.112 ± 0.008	0.478 ± 0.140	0.238 ± 0.005	**0.059 ± 0.063**
euclidEF300	HV	0.822 ± 0.005	0.760 ± 0.036	0.288 ± 0.209	0.694 ± 0.016	**0.848 ± 0.053**
	IGD	0.087 ± 0.008	0.116 ± 0.012	0.519 ± 0.155	0.252 ± 0.006	**0.065 ± 0.073**
mixedGG300	HV	0.826 ± 0.011	0.810 ± 0.025	0.334 ± 0.216	0.701 ± 0.020	**0.882 ± 0.042**
	IGD	0.144 ± 0.005	0.116 ± 0.005	0.492 ± 0.133	0.273 ± 0.008	**0.082 ± 0.078**
mixedHH300	HV	0.823 ± 0.012	0.802 ± 0.029	0.344 ± 0.210	0.707 ± 0.020	**0.876 ± 0.047**
	IGD	0.155 ± 0.006	0.119 ± 0.007	0.481 ± 0.117	0.278 ± 0.012	**0.086 ± 0.084**
mixedII300	HV	0.820 ± 0.012	0.801 ± 0.034	0.322 ± 0.207	0.690 ± 0.025	**0.872 ± 0.048**
	IGD	0.137 ± 0.007	0.114 ± 0.004	0.491 ± 0.129	0.268 ± 0.008	**0.083 ± 0.084**
randomAB300	HV	0.831 ± 0.017	0.833 ± 0.017	0.346 ± 0.242	0.717 ± 0.026	**0.899 ± 0.037**
	IGD	0.129 ± 0.008	0.124 ± 0.019	0.509 ± 0.193	0.220 ± 0.001	**0.081 ± 0.061**
randomCD300	HV	0.828 ± 0.017	0.841 ± 0.017	0.338 ± 0.248	0.715 ± 0.024	**0.900 ± 0.036**
	IGD	0.134 ± 0.011	0.121 ± 0.025	0.525 ± 0.193	0.229 ± 0.007	**0.084 ± 0.063**
randomEF300	HV	0.830 ± 0.017	0.841 ± 0.019	0.345 ± 0.241	0.718 ± 0.033	**0.900 ± 0.039**
	IGD	0.133 ± 0.009	0.119 ± 0.020	0.502 ± 0.181	0.232 ± 0.003	**0.084 ± 0.072**
euclidAB500	HV	0.838 ± 0.006	0.783 ± 0.032	0.295 ± 0.223	0.721 ± 0.015	**0.873 ± 0.052**
	IGD	0.108 ± 0.009	0.121 ± 0.012	0.520 ± 0.161	0.257 ± 0.005	**0.070 ± 0.080**
euclidCD500	HV	0.835 ± 0.005	0.777 ± 0.034	0.290 ± 0.217	0.717 ± 0.018	**0.870 ± 0.050**
	IGD	0.103 ± 0.008	0.119 ± 0.007	0.521 ± 0.155	0.255 ± 0.002	**0.067 ± 0.076**
euclidEF500	HV	0.833 ± 0.005	0.773 ± 0.032	0.278 ± 0.218	0.711 ± 0.016	**0.866 ± 0.050**
	IGD	0.108 ± 0.009	0.120 ± 0.009	0.525 ± 0.156	0.260 ± 0.003	**0.068 ± 0.074**
mixedGG500	HV	0.834 ± 0.010	0.805 ± 0.024	0.313 ± 0.221	0.608 ± 0.223	**0.892 ± 0.047**
	IGD	0.154 ± 0.007	0.127 ± 0.013	0.517 ± 0.145	0.329 ± 0.153	**0.085 ± 0.083**
mixedHH500	HV	0.834 ± 0.010	0.810 ± 0.025	0.310 ± 0.220	0.610 ± 0.224	**0.892 ± 0.047**
	IGD	0.162 ± 0.008	0.122 ± 0.012	0.515 ± 0.138	0.338 ± 0.142	**0.089 ± 0.094**
mixedII500	HV	0.837 ± 0.011	0.809 ± 0.024	0.303 ± 0.226	0.719 ± 0.022	**0.894 ± 0.046**
	IGD	0.157 ± 0.008	0.130 ± 0.012	0.526 ± 0.151	0.267 ± 0.005	**0.088 ± 0.089**
randomAB500	HV	0.839 ± 0.016	0.837 ± 0.021	0.308 ± 0.260	0.617 ± 0.217	**0.913 ± 0.040**
	IGD	0.140 ± 0.011	0.136 ± 0.024	0.563 ± 0.221	0.299 ± 0.175	**0.083 ± 0.069**
randomCD500	HV	0.840 ± 0.016	0.840 ± 0.022	0.305 ± 0.272	0.610 ± 0.219	**0.915 ± 0.037**
	IGD	0.144 ± 0.008	0.133 ± 0.019	0.569 ± 0.229	0.305 ± 0.172	**0.086 ± 0.067**
randomEF500	HV	0.846 ± 0.014	0.845 ± 0.026	0.333 ± 0.263	0.631 ± 0.210	**0.919 ± 0.034**
	IGD	0.136 ± 0.010	0.122 ± 0.015	0.543 ± 0.218	0.281 ± 0.158	**0.079 ± 0.058**

Table 17: Ablation experimental results of MOSAIC on Bi-KP. Higher HV and lower IGD indicate better performance. The best and second-best values are highlighted in bold and underlined, respectively. Results are averaged over five independent runs.

Instance	Metric	MOSAIC w/o AR	MOSAIC w/o BR	MOSAIC w/o AR+BR	Fixed-Arch	MOSAIC
2KP200-1A	HV	0.568 ± 0.003	$\underline{0.582} \pm 0.030$	0.353 ± 0.063	0.565 ± 0.001	$\mathbf{0.595} \pm 0.015$
	IGD	0.225 ± 0.003	$\underline{0.202} \pm 0.033$	0.319 ± 0.056	0.229 ± 0.019	$\mathbf{0.199} \pm 0.013$
2KP200-1B	HV	0.629 ± 0.010	$\underline{0.635} \pm 0.005$	0.408 ± 0.037	0.626 ± 0.000	$\mathbf{0.658} \pm 0.009$
	IGD	0.208 ± 0.013	$\underline{0.202} \pm 0.011$	0.304 ± 0.030	0.229 ± 0.001	$\mathbf{0.182} \pm 0.021$
2KP200-1C	HV	0.569 ± 0.004	$\underline{0.591} \pm 0.013$	0.411 ± 0.014	0.579 ± 0.014	$\mathbf{0.593} \pm 0.016$
	IGD	$\mathbf{0.191} \pm 0.008$	$\underline{0.192} \pm 0.007$	0.271 ± 0.013	0.217 ± 0.010	0.197 ± 0.020
2KP200-1D	HV	0.586 ± 0.001	$\underline{0.618} \pm 0.005$	0.397 ± 0.030	0.594 ± 0.017	$\mathbf{0.619} \pm 0.006$
	IGD	0.217 ± 0.020	$\mathbf{0.196} \pm 0.007$	0.295 ± 0.026	0.221 ± 0.004	$\underline{0.205} \pm 0.011$
2KP250-1A	HV	0.588 ± 0.003	$\underline{0.605} \pm 0.016$	0.323 ± 0.039	0.583 ± 0.009	$\mathbf{0.612} \pm 0.007$
	IGD	0.219 ± 0.008	$\underline{0.212} \pm 0.024$	0.362 ± 0.038	0.221 ± 0.001	$\mathbf{0.205} \pm 0.011$
2KP250-1B	HV	0.562 ± 0.002	0.574 ± 0.000	0.296 ± 0.048	$\underline{0.578} \pm 0.005$	$\mathbf{0.594} \pm 0.009$
	IGD	$\underline{0.215} \pm 0.006$	0.216 ± 0.008	0.376 ± 0.053	0.223 ± 0.015	$\mathbf{0.203} \pm 0.021$
2KP250-1C	HV	0.532 ± 0.005	$\underline{0.547} \pm 0.005$	0.296 ± 0.034	0.530 ± 0.012	$\mathbf{0.558} \pm 0.006$
	IGD	0.268 ± 0.020	$\mathbf{0.234} \pm 0.019$	0.380 ± 0.035	0.280 ± 0.019	$\underline{0.251} \pm 0.007$
2KP250-1D	HV	0.614 ± 0.002	$\underline{0.635} \pm 0.013$	0.233 ± 0.066	0.602 ± 0.002	$\mathbf{0.638} \pm 0.009$
	IGD	$\underline{0.202} \pm 0.002$	$\mathbf{0.173} \pm 0.010$	0.471 ± 0.080	0.236 ± 0.010	0.203 ± 0.013
2KP300-1A	HV	$\underline{0.590} \pm 0.000$	0.585 ± 0.001	0.190 ± 0.058	0.588 ± 0.010	$\mathbf{0.603} \pm 0.013$
	IGD	0.217 ± 0.001	$\mathbf{0.201} \pm 0.021$	0.523 ± 0.089	0.226 ± 0.006	$\underline{0.208} \pm 0.016$
2KP300-1B	HV	0.582 ± 0.004	0.576 ± 0.007	0.158 ± 0.064	$\underline{0.584} \pm 0.011$	$\mathbf{0.593} \pm 0.010$
	IGD	$\underline{0.220} \pm 0.005$	0.231 ± 0.028	0.552 ± 0.100	$\underline{0.220} \pm 0.021$	$\mathbf{0.209} \pm 0.019$
2KP300-1C	HV	0.571 ± 0.006	$\mathbf{0.593} \pm 0.001$	0.157 ± 0.048	0.568 ± 0.001	$\underline{0.591} \pm 0.009$
	IGD	0.250 ± 0.001	$\mathbf{0.228} \pm 0.004$	0.559 ± 0.073	0.264 ± 0.007	$\underline{0.237} \pm 0.008$
2KP300-1D	HV	$\underline{0.581} \pm 0.008$	0.551 ± 0.016	0.001 ± 0.001	0.566 ± 0.006	$\mathbf{0.594} \pm 0.013$
	IGD	0.235 ± 0.012	$\underline{0.224} \pm 0.022$	1.097 ± 0.129	0.258 ± 0.004	$\mathbf{0.219} \pm 0.013$
2KP350-1A	HV	0.587 ± 0.002	0.585 ± 0.004	0.135 ± 0.061	$\underline{0.593} \pm 0.003$	$\mathbf{0.601} \pm 0.013$
	IGD	$\underline{0.224} \pm 0.001$	$\mathbf{0.221} \pm 0.007$	0.607 ± 0.120	0.227 ± 0.005	0.225 ± 0.010
2KP350-1B	HV	0.577 ± 0.006	0.579 ± 0.004	0.184 ± 0.072	$\underline{0.580} \pm 0.004$	$\mathbf{0.593} \pm 0.006$
	IGD	0.237 ± 0.002	$\underline{0.235} \pm 0.002$	0.526 ± 0.100	0.246 ± 0.002	$\mathbf{0.228} \pm 0.009$
2KP350-1C	HV	0.569 ± 0.002	$\underline{0.589} \pm 0.005$	0.217 ± 0.048	0.582 ± 0.026	$\mathbf{0.596} \pm 0.012$
	IGD	0.249 ± 0.009	$\mathbf{0.223} \pm 0.010$	0.479 ± 0.063	0.245 ± 0.032	$\mathbf{0.223} \pm 0.010$
2KP350-1D	HV	$\underline{0.597} \pm 0.002$	0.587 ± 0.013	0.031 ± 0.037	0.583 ± 0.007	$\mathbf{0.605} \pm 0.014$
	IGD	$\mathbf{0.218} \pm 0.026$	$\underline{0.209} \pm 0.023$	0.865 ± 0.151	0.247 ± 0.009	0.223 ± 0.015
2KP400-1A	HV	0.567 ± 0.007	0.562 ± 0.004	0.108 ± 0.044	$\underline{0.571} \pm 0.004$	$\mathbf{0.585} \pm 0.007$
	IGD	$\underline{0.236} \pm 0.011$	0.238 ± 0.011	0.649 ± 0.083	0.249 ± 0.001	$\mathbf{0.229} \pm 0.009$
2KP400-1B	HV	0.584 ± 0.006	0.582 ± 0.001	0.085 ± 0.037	$\underline{0.586} \pm 0.013$	$\mathbf{0.596} \pm 0.008$
	IGD	0.217 ± 0.000	$\underline{0.211} \pm 0.010$	0.697 ± 0.080	0.235 ± 0.015	$\mathbf{0.207} \pm 0.010$
2KP400-1C	HV	0.569 ± 0.000	0.567 ± 0.003	0.138 ± 0.059	$\underline{0.571} \pm 0.003$	$\mathbf{0.585} \pm 0.007$
	IGD	0.253 ± 0.002	$\mathbf{0.227} \pm 0.008$	0.593 ± 0.093	0.255 ± 0.005	$\underline{0.241} \pm 0.012$
2KP400-1D	HV	0.573 ± 0.003	$\underline{0.583} \pm 0.003$	0.329 ± 0.034	0.568 ± 0.007	$\mathbf{0.595} \pm 0.004$
	IGD	0.247 ± 0.004	$\underline{0.230} \pm 0.011$	0.354 ± 0.029	0.257 ± 0.007	$\mathbf{0.221} \pm 0.009$
2KP450-1A	HV	0.560 ± 0.004	$\underline{0.561} \pm 0.002$	0.113 ± 0.056	0.556 ± 0.002	$\mathbf{0.572} \pm 0.004$
	IGD	0.229 ± 0.010	$\mathbf{0.208} \pm 0.015$	0.627 ± 0.100	0.243 ± 0.001	$\underline{0.216} \pm 0.012$
2KP450-1B	HV	0.541 ± 0.002	$\underline{0.553} \pm 0.005$	0.174 ± 0.040	0.538 ± 0.009	$\mathbf{0.563} \pm 0.005$
	IGD	0.235 ± 0.006	$\mathbf{0.204} \pm 0.006$	0.522 ± 0.065	0.242 ± 0.009	$\underline{0.217} \pm 0.009$
2KP450-1C	HV	0.567 ± 0.001	0.562 ± 0.014	0.012 ± 0.015	$\underline{0.574} \pm 0.001$	$\mathbf{0.577} \pm 0.009$
	IGD	0.240 ± 0.011	$\mathbf{0.221} \pm 0.012$	0.890 ± 0.066	0.239 ± 0.001	$\mathbf{0.221} \pm 0.016$
2KP450-1D	HV	$\underline{0.543} \pm 0.006$	0.500 ± 0.020	0.000 ± 0.000	$\mathbf{0.555} \pm 0.006$	0.545 ± 0.016
	IGD	$\underline{0.214} \pm 0.005$	0.226 ± 0.034	1.446 ± 0.100	0.215 ± 0.012	$\mathbf{0.212} \pm 0.016$
2KP500-1A	HV	0.579 ± 0.001	0.570 ± 0.001	0.078 ± 0.028	$\underline{0.584} \pm 0.003$	$\mathbf{0.590} \pm 0.006$
	IGD	0.239 ± 0.011	$\mathbf{0.228} \pm 0.008$	0.700 ± 0.059	0.240 ± 0.000	$\underline{0.230} \pm 0.014$
2KP500-1B	HV	0.540 ± 0.000	0.530 ± 0.014	0.076 ± 0.037	$\underline{0.544} \pm 0.002$	$\mathbf{0.556} \pm 0.009$
	IGD	0.248 ± 0.003	$\underline{0.229} \pm 0.010$	0.685 ± 0.080	0.249 ± 0.001	$\mathbf{0.227} \pm 0.013$
2KP500-1C	HV	$\underline{0.551} \pm 0.001$	0.536 ± 0.025	0.000 ± 0.000	0.550 ± 0.002	$\mathbf{0.565} \pm 0.006$
	IGD	$\underline{0.223} \pm 0.003$	$\underline{0.223} \pm 0.015$	1.232 ± 0.129	0.245 ± 0.005	$\mathbf{0.207} \pm 0.023$
2KP500-1D	HV	$\underline{0.574} \pm 0.002$	0.557 ± 0.011	0.004 ± 0.009	0.569 ± 0.007	$\mathbf{0.586} \pm 0.007$
	IGD	0.230 ± 0.013	$\underline{0.229} \pm 0.002$	0.999 ± 0.101	0.254 ± 0.006	$\mathbf{0.227} \pm 0.011$

Table 18: Ablation experimental results of MOSAIC on Bi-GVRP. Higher HV and lower IGD indicate better performance. The best and second-best values are highlighted in bold and underlined, respectively. Results are averaged over five independent runs.

Instance	Metric	MOSAIC w/o AR	MOSAIC w/o BR	MOSAIC w/o AR+BR	Fixed-Arch	MOSAIC
CMT11	HV	$\underline{0.796} \pm 0.004$	0.721 ± 0.004	0.527 ± 0.005	0.640 ± 0.018	$\mathbf{0.816} \pm 0.021$
	IGD	$\underline{0.050} \pm 0.003$	0.111 ± 0.003	0.286 ± 0.005	0.181 ± 0.017	$\mathbf{0.033} \pm 0.017$
CMT12	HV	$\underline{0.643} \pm 0.006$	0.554 ± 0.015	0.393 ± 0.026	0.479 ± 0.041	$\mathbf{0.695} \pm 0.043$
	IGD	$\underline{0.110} \pm 0.003$	0.193 ± 0.012	0.357 ± 0.029	0.266 ± 0.046	$\mathbf{0.063} \pm 0.037$
CMT4	HV	0.350 ± 0.003	0.342 ± 0.003	0.306 ± 0.040	$\underline{0.436} \pm 0.017$	$\mathbf{0.579} \pm 0.199$
	IGD	0.414 ± 0.001	0.418 ± 0.004	0.463 ± 0.049	$\underline{0.312} \pm 0.019$	$\mathbf{0.182} \pm 0.194$
CMT5	HV	0.327 ± 0.003	0.295 ± 0.006	0.259 ± 0.056	$\underline{0.395} \pm 0.019$	$\mathbf{0.573} \pm 0.220$
	IGD	0.440 ± 0.002	0.483 ± 0.013	0.531 ± 0.079	$\underline{0.360} \pm 0.020$	$\mathbf{0.195} \pm 0.217$
Golden_10	HV	$\underline{0.792} \pm 0.000$	0.721 ± 0.036	0.557 ± 0.018	0.601 ± 0.012	$\mathbf{0.840} \pm 0.045$
	IGD	$\underline{0.080} \pm 0.000$	0.139 ± 0.030	0.288 ± 0.019	0.243 ± 0.011	$\mathbf{0.042} \pm 0.035$
Golden_11	HV	$\underline{0.809} \pm 0.000$	0.734 ± 0.039	0.556 ± 0.039	0.566 ± 0.026	$\mathbf{0.854} \pm 0.042$
	IGD	$\underline{0.074} \pm 0.000$	0.135 ± 0.033	0.295 ± 0.037	0.283 ± 0.025	$\mathbf{0.039} \pm 0.032$
Golden_12	HV	$\underline{0.824} \pm 0.000$	0.769 ± 0.004	0.544 ± 0.019	0.563 ± 0.019	$\mathbf{0.865} \pm 0.039$
	IGD	$\underline{0.068} \pm 0.000$	0.112 ± 0.003	0.312 ± 0.019	0.293 ± 0.019	$\mathbf{0.036} \pm 0.030$
Golden_14	HV	$\mathbf{0.772} \pm 0.001$	0.666 ± 0.036	0.263 ± 0.004	0.354 ± 0.011	$\underline{0.764} \pm 0.014$
	IGD	$\mathbf{0.020} \pm 0.001$	0.110 ± 0.031	0.536 ± 0.005	0.423 ± 0.015	$\underline{0.027} \pm 0.012$
Golden_15	HV	$\mathbf{0.793} \pm 0.001$	0.718 ± 0.023	0.266 ± 0.018	0.322 ± 0.023	$\underline{0.786} \pm 0.012$
	IGD	$\mathbf{0.017} \pm 0.000$	0.080 ± 0.019	0.547 ± 0.023	0.476 ± 0.028	$\underline{0.023} \pm 0.010$
Golden_16	HV	$\mathbf{0.810} \pm 0.000$	0.742 ± 0.007	0.260 ± 0.027	0.300 ± 0.001	$\underline{0.802} \pm 0.012$
	IGD	$\mathbf{0.017} \pm 0.000$	0.072 ± 0.007	0.568 ± 0.037	0.514 ± 0.001	$\underline{0.022} \pm 0.010$
Golden_18	HV	$\underline{0.489} \pm 0.003$	0.419 ± 0.004	0.235 ± 0.026	0.323 ± 0.007	$\mathbf{0.633} \pm 0.134$
	IGD	$\underline{0.252} \pm 0.004$	0.326 ± 0.005	0.552 ± 0.037	0.436 ± 0.008	$\mathbf{0.121} \pm 0.123$
Golden_19	HV	$\underline{0.487} \pm 0.002$	0.411 ± 0.018	0.221 ± 0.007	0.317 ± 0.027	$\mathbf{0.633} \pm 0.135$
	IGD	$\underline{0.253} \pm 0.002$	0.334 ± 0.020	0.572 ± 0.010	0.444 ± 0.033	$\mathbf{0.120} \pm 0.124$
Golden_2	HV	$\mathbf{0.797} \pm 0.000$	0.709 ± 0.055	0.205 ± 0.002	0.343 ± 0.016	$\underline{0.760} \pm 0.088$
	IGD	$\mathbf{0.037} \pm 0.000$	0.112 ± 0.048	0.652 ± 0.003	0.477 ± 0.018	$\underline{0.070} \pm 0.078$
Golden_20	HV	$\underline{0.488} \pm 0.003$	0.415 ± 0.004	0.203 ± 0.009	0.282 ± 0.019	$\mathbf{0.633} \pm 0.137$
	IGD	$\underline{0.253} \pm 0.003$	0.331 ± 0.003	0.601 ± 0.016	0.488 ± 0.025	$\mathbf{0.121} \pm 0.125$
Golden_4	HV	$\mathbf{0.820} \pm 0.000$	0.755 ± 0.029	0.204 ± 0.005	0.293 ± 0.019	$\underline{0.799} \pm 0.079$
	IGD	$\mathbf{0.049} \pm 0.000$	0.103 ± 0.025	0.683 ± 0.006	0.567 ± 0.024	$\underline{0.067} \pm 0.068$
Golden_6	HV	$\underline{0.763} \pm 0.000$	0.667 ± 0.048	0.237 ± 0.023	0.382 ± 0.011	$\mathbf{0.767} \pm 0.065$
	IGD	$\underline{0.073} \pm 0.000$	0.157 ± 0.043	0.617 ± 0.031	0.440 ± 0.012	$\mathbf{0.072} \pm 0.056$
Golden_7	HV	$\mathbf{0.800} \pm 0.000$	0.625 ± 0.054	0.217 ± 0.011	0.334 ± 0.011	$\underline{0.782} \pm 0.075$
	IGD	$\mathbf{0.050} \pm 0.000$	0.202 ± 0.050	0.650 ± 0.016	0.502 ± 0.013	$\underline{0.066} \pm 0.066$
Golden_8	HV	$\mathbf{0.825} \pm 0.000$	0.691 ± 0.003	0.207 ± 0.002	0.307 ± 0.004	$\underline{0.785} \pm 0.086$
	IGD	$\mathbf{0.035} \pm 0.000$	0.147 ± 0.003	0.668 ± 0.002	0.540 ± 0.005	$\underline{0.069} \pm 0.075$

C.7 Detailed Results across Different LLMs

Tables 19–21 report detailed per-instance HV and IGD results of MOSAIC with different LLMs on Bi-TSP, Bi-KP, and Bi-GVRP. Results are reported as the mean and standard deviation over five independent runs.

Across the three problems, MOSAIC maintains strong performance across the tested LLMs, although the relative performance varies across models, instances, and metrics. The overall performance patterns remain consistent, further indicating that MOSAIC is not strongly dependent on the choice of LLM. Considering both performance and API cost, we use DeepSeek-V4 Flash as the default LLM in the main experiments.

Table 19: Per-instance performance of MOSAIC with different LLMs under NSGA-II on the Bi-TSP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. Best and second-best means are shown in bold and underlined, respectively.

Instance	Metric	DeepSeek-v4 Flash	DeepSeek-v4 Pro	GLM-5.3	GPT-5.6 Sol
kroAB100	HV	0.857 ± 0.041	0.857 ± 0.042	$\mathbf{0.866} \pm 0.049$	$\underline{0.861} \pm 0.052$
	IGD	0.062 ± 0.052	0.061 ± 0.067	$\underline{0.057} \pm 0.070$	$\mathbf{0.049} \pm 0.072$
kroAC100	HV	0.852 ± 0.044	0.853 ± 0.048	$\underline{0.860} \pm 0.054$	$\mathbf{0.865} \pm 0.050$
	IGD	$\underline{0.053} \pm 0.047$	$\mathbf{0.048} \pm 0.063$	$\mathbf{0.048} \pm 0.065$	$\mathbf{0.048} \pm 0.054$
kroAD100	HV	0.837 ± 0.046	$\underline{0.840} \pm 0.054$	0.839 ± 0.053	$\mathbf{0.842} \pm 0.041$
	IGD	0.052 ± 0.049	$\underline{0.047} \pm 0.073$	0.049 ± 0.058	$\mathbf{0.041} \pm 0.055$
kroAE100	HV	0.853 ± 0.045	$\underline{0.854} \pm 0.053$	$\mathbf{0.856} \pm 0.046$	$\underline{0.854} \pm 0.048$
	IGD	0.055 ± 0.054	$\underline{0.053} \pm 0.072$	0.054 ± 0.064	$\mathbf{0.051} \pm 0.062$
kroBC100	HV	0.842 ± 0.048	$\underline{0.845} \pm 0.051$	$\underline{0.845} \pm 0.049$	$\mathbf{0.855} \pm 0.052$
	IGD	0.054 ± 0.054	0.053 ± 0.056	$\underline{0.052} \pm 0.053$	$\mathbf{0.045} \pm 0.051$
kroBD100	HV	0.837 ± 0.040	0.837 ± 0.042	$\underline{0.845} \pm 0.040$	$\mathbf{0.846} \pm 0.049$
	IGD	0.049 ± 0.042	0.045 ± 0.067	$\underline{0.041} \pm 0.053$	$\mathbf{0.036} \pm 0.069$
kroBE100	HV	0.846 ± 0.044	0.850 ± 0.051	$\mathbf{0.852} \pm 0.041$	$\underline{0.851} \pm 0.043$
	IGD	0.054 ± 0.049	0.053 ± 0.067	$\underline{0.050} \pm 0.074$	$\mathbf{0.040} \pm 0.064$
kroCD100	HV	0.850 ± 0.044	$\underline{0.854} \pm 0.044$	$\underline{0.854} \pm 0.053$	$\mathbf{0.859} \pm 0.041$
	IGD	0.056 ± 0.050	0.056 ± 0.057	$\underline{0.054} \pm 0.070$	$\mathbf{0.042} \pm 0.050$
kroCE100	HV	0.850 ± 0.045	$\underline{0.852} \pm 0.053$	$\underline{0.852} \pm 0.047$	$\mathbf{0.864} \pm 0.052$
	IGD	0.057 ± 0.057	$\underline{0.054} \pm 0.068$	0.057 ± 0.060	$\mathbf{0.046} \pm 0.072$
kroDE100	HV	0.848 ± 0.045	0.850 ± 0.054	$\underline{0.854} \pm 0.048$	$\mathbf{0.855} \pm 0.045$
	IGD	0.049 ± 0.047	0.045 ± 0.069	$\underline{0.040} \pm 0.054$	$\mathbf{0.035} \pm 0.058$
kroAB150	HV	0.859 ± 0.043	$\underline{0.863} \pm 0.047$	0.859 ± 0.048	$\mathbf{0.870} \pm 0.045$
	IGD	0.056 ± 0.055	$\underline{0.052} \pm 0.066$	$\mathbf{0.050} \pm 0.063$	$\mathbf{0.050} \pm 0.073$
kroAB200	HV	0.863 ± 0.046	0.866 ± 0.045	$\underline{0.872} \pm 0.045$	$\mathbf{0.873} \pm 0.043$
	IGD	0.058 ± 0.053	0.057 ± 0.057	$\underline{0.050} \pm 0.072$	$\mathbf{0.047} \pm 0.058$
euclidAB300	HV	0.851 ± 0.052	$\underline{0.854} \pm 0.051$	$\underline{0.854} \pm 0.045$	$\mathbf{0.856} \pm 0.052$
	IGD	0.067 ± 0.079	0.066 ± 0.070	$\underline{0.063} \pm 0.054$	$\mathbf{0.061} \pm 0.063$
euclidCD300	HV	0.854 ± 0.049	$\underline{0.856} \pm 0.045$	$\underline{0.856} \pm 0.055$	$\mathbf{0.861} \pm 0.042$
	IGD	0.059 ± 0.063	0.058 ± 0.051	$\mathbf{0.055} \pm 0.072$	$\underline{0.057} \pm 0.054$
euclidEF300	HV	0.848 ± 0.053	0.850 ± 0.041	$\underline{0.857} \pm 0.054$	$\mathbf{0.862} \pm 0.052$
	IGD	0.065 ± 0.073	0.062 ± 0.066	$\mathbf{0.058} \pm 0.072$	$\underline{0.060} \pm 0.058$
mixedGG300	HV	0.882 ± 0.042	$\mathbf{0.886} \pm 0.055$	$\underline{0.884} \pm 0.043$	0.883 ± 0.043
	IGD	0.082 ± 0.078	0.082 ± 0.058	$\underline{0.078} \pm 0.068$	$\mathbf{0.069} \pm 0.058$
mixedHH300	HV	0.876 ± 0.047	$\underline{0.880} \pm 0.045$	0.877 ± 0.046	$\mathbf{0.887} \pm 0.047$
	IGD	0.086 ± 0.084	0.085 ± 0.074	$\underline{0.084} \pm 0.056$	$\mathbf{0.075} \pm 0.061$
mixedII300	HV	0.872 ± 0.048	0.873 ± 0.053	$\mathbf{0.880} \pm 0.040$	$\underline{0.879} \pm 0.043$
	IGD	0.083 ± 0.084	0.079 ± 0.057	$\underline{0.073} \pm 0.068$	$\mathbf{0.070} \pm 0.056$
randomAB300	HV	0.899 ± 0.037	$\underline{0.902} \pm 0.054$	$\mathbf{0.904} \pm 0.044$	$\underline{0.902} \pm 0.041$
	IGD	0.081 ± 0.061	$\underline{0.079} \pm 0.058$	$\underline{0.079} \pm 0.062$	$\mathbf{0.073} \pm 0.060$
randomCD300	HV	0.900 ± 0.036	0.902 ± 0.055	$\underline{0.909} \pm 0.050$	$\mathbf{0.914} \pm 0.043$
	IGD	0.084 ± 0.063	0.082 ± 0.051	$\underline{0.078} \pm 0.050$	$\mathbf{0.077} \pm 0.074$
randomEF300	HV	0.900 ± 0.039	0.903 ± 0.044	$\underline{0.910} \pm 0.046$	$\mathbf{0.911} \pm 0.048$
	IGD	0.084 ± 0.072	$\underline{0.082} \pm 0.070$	$\mathbf{0.079} \pm 0.059$	0.084 ± 0.052
euclidAB500	HV	0.873 ± 0.052	0.874 ± 0.052	$\underline{0.876} \pm 0.050$	$\mathbf{0.882} \pm 0.046$
	IGD	0.070 ± 0.080	$\underline{0.066} \pm 0.071$	$\underline{0.066} \pm 0.059$	$\mathbf{0.055} \pm 0.061$
euclidCD500	HV	0.870 ± 0.050	0.872 ± 0.049	$\underline{0.874} \pm 0.053$	$\mathbf{0.880} \pm 0.055$
	IGD	0.067 ± 0.076	$\underline{0.065} \pm 0.066$	0.066 ± 0.053	$\mathbf{0.052} \pm 0.067$
euclidEF500	HV	0.866 ± 0.050	0.869 ± 0.053	$\mathbf{0.875} \pm 0.041$	$\underline{0.871} \pm 0.043$
	IGD	0.068 ± 0.074	0.067 ± 0.073	$\mathbf{0.061} \pm 0.069$	$\underline{0.065} \pm 0.071$
mixedGG500	HV	0.892 ± 0.047	$\mathbf{0.894} \pm 0.051$	$\mathbf{0.894} \pm 0.052$	$\underline{0.893} \pm 0.046$
	IGD	0.085 ± 0.083	0.085 ± 0.064	$\underline{0.079} \pm 0.054$	$\mathbf{0.072} \pm 0.061$
mixedHH500	HV	0.892 ± 0.047	0.894 ± 0.047	$\mathbf{0.901} \pm 0.053$	$\underline{0.895} \pm 0.046$
	IGD	0.089 ± 0.094	$\underline{0.086} \pm 0.070$	0.088 ± 0.057	$\mathbf{0.078} \pm 0.051$
mixedII500	HV	0.894 ± 0.046	$\underline{0.899} \pm 0.048$	0.896 ± 0.053	$\mathbf{0.908} \pm 0.048$
	IGD	0.088 ± 0.089	0.088 ± 0.065	$\mathbf{0.082} \pm 0.068$	$\underline{0.085} \pm 0.072$
randomAB500	HV	0.913 ± 0.040	0.914 ± 0.045	$\underline{0.918} \pm 0.047$	$\mathbf{0.927} \pm 0.047$
	IGD	0.083 ± 0.069	0.081 ± 0.054	$\mathbf{0.074} \pm 0.073$	$\underline{0.077} \pm 0.069$
randomCD500	HV	0.915 ± 0.037	0.916 ± 0.049	$\underline{0.919} \pm 0.053$	$\mathbf{0.928} \pm 0.048$
	IGD	0.086 ± 0.067	0.085 ± 0.073	$\underline{0.080} \pm 0.066$	$\mathbf{0.075} \pm 0.058$
randomEF500	HV	0.919 ± 0.034	0.920 ± 0.050	$\mathbf{0.928} \pm 0.054$	$\underline{0.924} \pm 0.050$
	IGD	0.079 ± 0.058	$\underline{0.077} \pm 0.065$	$\mathbf{0.073} \pm 0.070$	0.079 ± 0.074

Table 20: Per-instance performance of MOSAIC with different LLMs under NSGA-II on the Bi-KP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. Best and second-best means are shown in bold and underlined, respectively.

Instance	Metric	DeepSeek-v4 Flash	DeepSeek-v4 Pro	GLM-5.3	GPT-5.6 Sol
2KP200-1A	HV	0.595 ± 0.015	0.595 ± 0.014	$\mathbf{0.602} \pm 0.019$	$\underline{0.598} \pm 0.021$
	IGD	0.199 ± 0.013	0.198 ± 0.018	$\underline{0.195} \pm 0.019$	$\mathbf{0.188} \pm 0.020$
2KP200-1B	HV	0.658 ± 0.009	0.659 ± 0.013	$\underline{0.664} \pm 0.017$	$\mathbf{0.668} \pm 0.014$
	IGD	0.182 ± 0.021	$\mathbf{0.177} \pm 0.024$	$\underline{0.178} \pm 0.025$	$\underline{0.178} \pm 0.021$
2KP200-1C	HV	0.593 ± 0.016	$\underline{0.596} \pm 0.023$	0.595 ± 0.022	$\mathbf{0.597} \pm 0.015$
	IGD	0.197 ± 0.020	$\underline{0.192} \pm 0.027$	0.194 ± 0.021	$\mathbf{0.188} \pm 0.020$
2KP200-1D	HV	0.619 ± 0.006	$\underline{0.620} \pm 0.012$	$\mathbf{0.621} \pm 0.008$	$\underline{0.620} \pm 0.009$
	IGD	0.205 ± 0.011	$\underline{0.203} \pm 0.018$	0.204 ± 0.015	$\mathbf{0.201} \pm 0.014$
2KP250-1A	HV	0.612 ± 0.007	$\underline{0.615} \pm 0.012$	0.614 ± 0.011	$\mathbf{0.622} \pm 0.013$
	IGD	0.205 ± 0.011	$\underline{0.204} \pm 0.012$	$\underline{0.204} \pm 0.010$	$\mathbf{0.198} \pm 0.009$
2KP250-1B	HV	$\underline{0.594} \pm 0.009$	$\underline{0.594} \pm 0.008$	$\mathbf{0.601} \pm 0.007$	$\mathbf{0.601} \pm 0.013$
	IGD	0.203 ± 0.021	0.199 ± 0.026	$\underline{0.197} \pm 0.020$	$\mathbf{0.193} \pm 0.026$
2KP250-1C	HV	0.558 ± 0.006	$\underline{0.562} \pm 0.012$	$\mathbf{0.563} \pm 0.005$	$\underline{0.562} \pm 0.006$
	IGD	0.251 ± 0.007	0.250 ± 0.012	$\underline{0.248} \pm 0.014$	$\mathbf{0.240} \pm 0.011$
2KP250-1D	HV	0.638 ± 0.009	$\underline{0.642} \pm 0.009$	0.641 ± 0.016	$\mathbf{0.645} \pm 0.008$
	IGD	0.203 ± 0.013	0.203 ± 0.014	$\underline{0.201} \pm 0.019$	$\mathbf{0.192} \pm 0.011$
2KP300-1A	HV	0.603 ± 0.013	$\underline{0.605} \pm 0.020$	0.604 ± 0.016	$\mathbf{0.614} \pm 0.019$
	IGD	0.208 ± 0.016	$\underline{0.205} \pm 0.021$	0.208 ± 0.018	$\mathbf{0.199} \pm 0.023$
2KP300-1B	HV	0.593 ± 0.010	0.595 ± 0.017	$\underline{0.598} \pm 0.013$	$\mathbf{0.599} \pm 0.011$
	IGD	0.209 ± 0.019	0.205 ± 0.025	$\underline{0.202} \pm 0.019$	$\mathbf{0.198} \pm 0.020$
2KP300-1C	HV	0.591 ± 0.009	$\underline{0.595} \pm 0.012$	0.591 ± 0.012	$\mathbf{0.600} \pm 0.011$
	IGD	0.237 ± 0.008	$\underline{0.233} \pm 0.012$	$\mathbf{0.232} \pm 0.011$	$\mathbf{0.232} \pm 0.015$
2KP300-1D	HV	0.594 ± 0.013	0.597 ± 0.014	$\underline{0.601} \pm 0.014$	$\mathbf{0.602} \pm 0.013$
	IGD	0.219 ± 0.013	0.218 ± 0.014	$\underline{0.213} \pm 0.020$	$\mathbf{0.210} \pm 0.014$
2KP350-1A	HV	0.601 ± 0.013	$\underline{0.604} \pm 0.019$	0.603 ± 0.015	$\mathbf{0.605} \pm 0.019$
	IGD	0.225 ± 0.010	0.224 ± 0.016	$\underline{0.222} \pm 0.010$	$\mathbf{0.220} \pm 0.013$
2KP350-1B	HV	0.593 ± 0.006	$\underline{0.595} \pm 0.007$	$\underline{0.595} \pm 0.014$	$\mathbf{0.599} \pm 0.005$
	IGD	0.228 ± 0.009	0.227 ± 0.008	$\mathbf{0.225} \pm 0.016$	$\underline{0.226} \pm 0.009$
2KP350-1C	HV	0.596 ± 0.012	0.598 ± 0.011	$\underline{0.603} \pm 0.019$	$\mathbf{0.608} \pm 0.018$
	IGD	0.223 ± 0.010	0.220 ± 0.014	$\mathbf{0.218} \pm 0.017$	$\underline{0.219} \pm 0.011$
2KP350-1D	HV	0.605 ± 0.014	$\mathbf{0.609} \pm 0.022$	$\underline{0.606} \pm 0.014$	0.605 ± 0.014
	IGD	0.223 ± 0.015	0.223 ± 0.016	$\underline{0.219} \pm 0.020$	$\mathbf{0.213} \pm 0.016$
2KP400-1A	HV	0.585 ± 0.007	$\underline{0.589} \pm 0.008$	0.586 ± 0.009	$\mathbf{0.594} \pm 0.010$
	IGD	0.229 ± 0.009	$\underline{0.228} \pm 0.017$	$\underline{0.228} \pm 0.009$	$\mathbf{0.220} \pm 0.011$
2KP400-1B	HV	0.596 ± 0.008	0.597 ± 0.014	$\mathbf{0.603} \pm 0.006$	$\underline{0.602} \pm 0.008$
	IGD	0.207 ± 0.010	0.203 ± 0.011	$\underline{0.199} \pm 0.015$	$\mathbf{0.197} \pm 0.010$
2KP400-1C	HV	0.585 ± 0.007	$\underline{0.588} \pm 0.014$	$\mathbf{0.589} \pm 0.008$	0.587 ± 0.006
	IGD	0.241 ± 0.012	$\underline{0.239} \pm 0.013$	$\underline{0.239} \pm 0.015$	$\mathbf{0.235} \pm 0.014$
2KP400-1D	HV	0.595 ± 0.004	0.597 ± 0.012	$\underline{0.602} \pm 0.009$	$\mathbf{0.606} \pm 0.004$
	IGD	0.221 ± 0.009	0.219 ± 0.008	$\underline{0.216} \pm 0.007$	$\mathbf{0.215} \pm 0.017$
2KP450-1A	HV	0.572 ± 0.004	0.575 ± 0.005	$\underline{0.580} \pm 0.006$	$\mathbf{0.581} \pm 0.007$
	IGD	0.216 ± 0.012	$\underline{0.214} \pm 0.018$	$\mathbf{0.212} \pm 0.013$	0.216 ± 0.011
2KP450-1B	HV	0.563 ± 0.005	0.564 ± 0.011	$\underline{0.565} \pm 0.009$	$\mathbf{0.570} \pm 0.007$
	IGD	0.217 ± 0.009	$\underline{0.213} \pm 0.015$	0.214 ± 0.011	$\mathbf{0.205} \pm 0.011$
2KP450-1C	HV	0.577 ± 0.009	0.579 ± 0.013	$\underline{0.580} \pm 0.016$	$\mathbf{0.585} \pm 0.017$
	IGD	0.221 ± 0.016	$\underline{0.219} \pm 0.020$	0.221 ± 0.015	$\mathbf{0.209} \pm 0.021$
2KP450-1D	HV	0.545 ± 0.016	0.548 ± 0.022	$\mathbf{0.552} \pm 0.015$	$\underline{0.549} \pm 0.016$
	IGD	0.212 ± 0.016	0.211 ± 0.023	$\mathbf{0.206} \pm 0.022$	$\underline{0.210} \pm 0.022$
2KP500-1A	HV	0.590 ± 0.006	$\mathbf{0.592} \pm 0.011$	$\mathbf{0.592} \pm 0.012$	$\underline{0.591} \pm 0.008$
	IGD	0.230 ± 0.014	0.230 ± 0.018	$\underline{0.226} \pm 0.014$	$\mathbf{0.219} \pm 0.016$
2KP500-1B	HV	0.556 ± 0.009	$\underline{0.558} \pm 0.012$	$\mathbf{0.563} \pm 0.016$	$\underline{0.558} \pm 0.011$
	IGD	0.227 ± 0.013	$\underline{0.224} \pm 0.019$	0.226 ± 0.014	$\mathbf{0.218} \pm 0.011$
2KP500-1C	HV	0.565 ± 0.006	$\underline{0.570} \pm 0.009$	0.567 ± 0.013	$\mathbf{0.576} \pm 0.010$
	IGD	0.207 ± 0.023	0.207 ± 0.027	$\mathbf{0.203} \pm 0.028$	$\underline{0.205} \pm 0.030$
2KP500-1D	HV	0.586 ± 0.007	0.587 ± 0.009	$\underline{0.590} \pm 0.010$	$\mathbf{0.598} \pm 0.010$
	IGD	0.227 ± 0.011	0.225 ± 0.011	$\mathbf{0.220} \pm 0.018$	$\underline{0.222} \pm 0.017$

Table 21: Per-instance performance of MOSAIC with different LLMs under NSGA-II on the Bi-GVRP test set. HV and IGD are reported as mean ± standard deviation over five independent runs. Best and second-best means are shown in bold and underlined, respectively.

Instance	Metric	DeepSeek-v4 Flash	DeepSeek-v4 Pro	GLM-5.3	GPT-5.6 Sol
CMT11	HV	0.816 ± 0.021	0.816 ± 0.014	$\underline{0.817} \pm 0.031$	$\mathbf{0.818} \pm 0.021$
	IGD	0.033 ± 0.017	$\underline{0.028} \pm 0.012$	0.029 ± 0.016	$\mathbf{0.026} \pm 0.022$
CMT12	HV	0.695 ± 0.043	0.696 ± 0.046	$\underline{0.697} \pm 0.040$	$\mathbf{0.707} \pm 0.035$
	IGD	0.063 ± 0.037	$\mathbf{0.057} \pm 0.030$	0.060 ± 0.044	$\underline{0.058} \pm 0.043$
CMT4	HV	0.579 ± 0.199	0.580 ± 0.192	$\underline{0.582} \pm 0.201$	$\mathbf{0.585} \pm 0.198$
	IGD	0.182 ± 0.194	$\underline{0.178} \pm 0.197$	$\mathbf{0.176} \pm 0.188$	0.181 ± 0.203
CMT5	HV	0.573 ± 0.220	$\underline{0.574} \pm 0.214$	0.573 ± 0.224	$\mathbf{0.579} \pm 0.224$
	IGD	0.195 ± 0.217	$\underline{0.192} \pm 0.226$	$\underline{0.192} \pm 0.219$	$\mathbf{0.189} \pm 0.216$
Golden_10	HV	0.840 ± 0.045	$\underline{0.843} \pm 0.051$	$\mathbf{0.849} \pm 0.050$	0.842 ± 0.050
	IGD	0.042 ± 0.035	$\mathbf{0.037} \pm 0.034$	$\underline{0.040} \pm 0.029$	$\mathbf{0.037} \pm 0.034$
Golden_11	HV	0.854 ± 0.042	0.856 ± 0.044	$\underline{0.862} \pm 0.037$	$\mathbf{0.863} \pm 0.036$
	IGD	0.039 ± 0.032	0.037 ± 0.032	$\mathbf{0.035} \pm 0.041$	$\underline{0.036} \pm 0.035$
Golden_12	HV	0.865 ± 0.039	0.866 ± 0.032	$\underline{0.868} \pm 0.042$	$\mathbf{0.873} \pm 0.041$
	IGD	0.036 ± 0.030	$\underline{0.032} \pm 0.033$	$\mathbf{0.029} \pm 0.023$	$\mathbf{0.029} \pm 0.022$
Golden_14	HV	0.764 ± 0.014	$\underline{0.768} \pm 0.020$	0.766 ± 0.013	$\mathbf{0.776} \pm 0.015$
	IGD	0.027 ± 0.012	$\underline{0.023} \pm 0.007$	$\mathbf{0.020} \pm 0.015$	$\mathbf{0.020} \pm 0.014$
Golden_15	HV	0.786 ± 0.012	0.789 ± 0.014	$\mathbf{0.795} \pm 0.006$	$\underline{0.794} \pm 0.005$
	IGD	0.023 ± 0.010	0.023 ± 0.004	$\mathbf{0.018} \pm 0.015$	$\underline{0.019} \pm 0.004$
Golden_16	HV	0.802 ± 0.012	0.803 ± 0.018	$\underline{0.807} \pm 0.016$	$\mathbf{0.809} \pm 0.021$
	IGD	0.022 ± 0.010	0.020 ± 0.011	$\underline{0.018} \pm 0.005$	$\mathbf{0.014} \pm 0.014$
Golden_18	HV	0.633 ± 0.134	0.634 ± 0.133	$\underline{0.636} \pm 0.141$	$\mathbf{0.641} \pm 0.133$
	IGD	0.121 ± 0.123	0.120 ± 0.116	$\underline{0.115} \pm 0.132$	$\mathbf{0.114} \pm 0.117$
Golden_19	HV	0.633 ± 0.135	0.634 ± 0.130	$\underline{0.638} \pm 0.145$	$\mathbf{0.643} \pm 0.144$
	IGD	0.120 ± 0.124	$\underline{0.118} \pm 0.132$	0.119 ± 0.120	$\mathbf{0.116} \pm 0.119$
Golden_2	HV	0.760 ± 0.088	$\mathbf{0.766} \pm 0.088$	$\underline{0.764} \pm 0.087$	0.760 ± 0.089
	IGD	0.070 ± 0.078	0.068 ± 0.080	$\mathbf{0.063} \pm 0.074$	$\underline{0.065} \pm 0.083$
Golden_20	HV	0.633 ± 0.137	0.634 ± 0.146	$\mathbf{0.639} \pm 0.132$	$\underline{0.635} \pm 0.143$
	IGD	0.121 ± 0.125	$\mathbf{0.119} \pm 0.119$	$\underline{0.120} \pm 0.131$	$\mathbf{0.119} \pm 0.126$
Golden_4	HV	0.799 ± 0.079	0.803 ± 0.080	$\underline{0.809} \pm 0.088$	$\mathbf{0.810} \pm 0.082$
	IGD	0.067 ± 0.068	$\mathbf{0.062} \pm 0.073$	0.066 ± 0.076	$\underline{0.065} \pm 0.077$
Golden_6	HV	0.767 ± 0.065	$\underline{0.771} \pm 0.059$	0.770 ± 0.061	$\mathbf{0.779} \pm 0.067$
	IGD	0.072 ± 0.056	$\underline{0.067} \pm 0.051$	$\mathbf{0.066} \pm 0.053$	0.070 ± 0.059
Golden_7	HV	0.782 ± 0.075	0.782 ± 0.075	$\mathbf{0.789} \pm 0.072$	$\underline{0.784} \pm 0.072$
	IGD	0.066 ± 0.066	$\underline{0.063} \pm 0.060$	$\mathbf{0.058} \pm 0.068$	$\underline{0.063} \pm 0.074$
Golden_8	HV	0.785 ± 0.086	$\mathbf{0.791} \pm 0.079$	$\underline{0.786} \pm 0.080$	$\mathbf{0.791} \pm 0.084$
	IGD	0.069 ± 0.075	$\underline{0.066} \pm 0.081$	$\underline{0.066} \pm 0.080$	$\mathbf{0.065} \pm 0.085$

Table 22: Performance of the 12 unique execution orders of the Bi-KP architecture. Results are reported as mean ± standard deviation over five independent runs, with ranks based on mean HV. The evolved architecture obtained during training is highlighted in bold.

Rank	Execution Order	HV	IGD
1	$O_{782} \rightarrow O_{103} \rightarrow O_{181} \rightarrow O_{181}$	0.590 ± 0.025	0.219 ± 0.020
2 (Evolved)	$\mathbf{O_{782} \rightarrow O_{181} \rightarrow O_{103} \rightarrow O_{181}}$	$\mathbf{0.585 \pm 0.026}$	$\mathbf{0.219 \pm 0.022}$
3	$O_{103} \rightarrow O_{782} \rightarrow O_{181} \rightarrow O_{181}$	0.577 ± 0.031	0.219 ± 0.024
4	$O_{181} \rightarrow O_{782} \rightarrow O_{103} \rightarrow O_{181}$	0.576 ± 0.028	0.222 ± 0.022
5	$O_{782} \rightarrow O_{181} \rightarrow O_{181} \rightarrow O_{103}$	0.567 ± 0.036	0.217 ± 0.023
6	$O_{181} \rightarrow O_{782} \rightarrow O_{181} \rightarrow O_{103}$	0.556 ± 0.039	0.223 ± 0.025
7	$O_{103} \rightarrow O_{181} \rightarrow O_{782} \rightarrow O_{181}$	0.550 ± 0.039	0.226 ± 0.022
8	$O_{181} \rightarrow O_{181} \rightarrow O_{782} \rightarrow O_{103}$	0.546 ± 0.043	0.226 ± 0.022
9	$O_{181} \rightarrow O_{103} \rightarrow O_{782} \rightarrow O_{181}$	0.543 ± 0.043	0.224 ± 0.024
10	$O_{103} \rightarrow O_{181} \rightarrow O_{181} \rightarrow O_{782}$	0.523 ± 0.050	0.232 ± 0.028
11	$O_{181} \rightarrow O_{103} \rightarrow O_{181} \rightarrow O_{782}$	0.513 ± 0.055	0.237 ± 0.029
12	$O_{181} \rightarrow O_{181} \rightarrow O_{103} \rightarrow O_{782}$	0.501 ± 0.055	0.238 ± 0.031

D Additional Discussion and Analysis

D.1 Additional Details on Module Synergy

This section provides additional details for the module synergy analysis in Section 4.1. The best architectures obtained during training on Bi-TSP, Bi-KP, and Bi-GVRP are $O_{634} \rightarrow O_{329}$, $O_{782} \rightarrow O_{181} \rightarrow O_{103} \rightarrow O_{181}$, and $O_{385} \rightarrow O_{491} \rightarrow O_{123} \rightarrow O_{201}$, respectively. These architectures illustrate that MOSAIC can identify problem-specific compositions and execution orders of search modules.

For Bi-KP, as discussed in Section 4.1, the evolved architecture combines cross-individual information recombination, population-guided adaptive mutation, and elite preservation. The repeated occurrence of O_{181} further indicates that architecture evolution can reuse the same module at different positions in the execution sequence. Together with the execution order analysis, this example illustrates that effective search depends not only on the behaviors implemented by individual modules but also on how these behaviors are organized. The executable code of the constituent modules in the three architectures is presented in Figs. 6–13.

We further isolate the effect of execution order by keeping the modules and their executable code unchanged and varying only their execution order. Since O_{181} appears twice in the four-module sequence, this yields 12 unique execution orders. All other experimental settings are identical to those used in the main experiments. Table 22 reports the complete results.

The evolved architecture ranks second in HV among the 12 execution orders and performs close to the best ordering. The results show clear performance variation when only the execution order is changed, with mean HV ranging from 0.501 to 0.590. All three orderings with O_{782} in the first position rank among the top five, whereas all three with O_{782} in the last position rank among the bottom three. IGD exhibits a consistent pattern, with the three orderings ending in O_{782} yielding the three highest IGD values. These results further support the order-dependent module synergy discussed in Section 4.1.

D.2 Additional Results on Large Module Pools

This section provides the complete per-instance results for the preliminary Bi-TSP study discussed in Section 4.2. Module rewriting is disabled, and only architecture evolution is performed over module pools of different sizes. All other experimental settings are identical to those used in the main experiments. The results are reported in Table 23.

Bi-TSP | **O634** → O392

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
(batch_size, city_count) = candidates.shape; dist1 = context.distance_matrix_1
    dist2 = context.distance_matrix_2;
    blend = 0.5 + 0.45 * np.sin(generation * 0.53 + 1.2)
    dist = dist1 * (1.0 - blend) + dist2 * blend;
    parent_pop = reference.parent_population
    sel_idx = reference.selected_indices;new_candidates = np.empty_like(candidates)
    noise_scale = 0.02 * np.exp(-generation / 30.0) + 0.003
    segment_prob = 0.35 + 0.25 * (generation % 4) / 3.0
    seed_parent_prob = 0.3 + 0.15 * np.cos(generation * 0.31)
    for i in range(batch_size):
        r = rng.random()
        if r < seed_parent_prob and len(sel_idx) > 0:
            p_idx = int(sel_idx[i % len(sel_idx)]);
            start_city = int(parent_pop[p_idx, 0])
        elif r < 0.5:
            start_city = int(rng.integers(0, city_count))
        else:
            start_city = int(candidates[i, 0])
        unvisited = np.ones(city_count, dtype=bool); unvisited[start_city] = False
        perm = [start_city]; current = start_city
        if rng.random() < segment_prob and len(sel_idx) > 0:
            p_idx = int(sel_idx[i % len(sel_idx)]); parent_perm = parent_pop[p_idx]
            pos = np.where(parent_perm == start_city)[0]
            if len(pos) > 0:
                pos = pos[0]; seg_len = min(2 + generation % 4, city_count - pos)
                if rng.random() < 0.6:
                    for k in range(1, seg_len):
                        nxt = int(parent_perm[pos + k])
                        if unvisited[nxt]:
                            perm.append(nxt); unvisited[nxt] = False
                            current = nxt
                        else:
                            break
                else:
                    start_pos = max(0, pos - seg_len + 1)
                    for k in range(pos - 1, start_pos - 1, -1):
                        nxt = int(parent_perm[k])
                        if unvisited[nxt]:
                            perm.append(nxt); unvisited[nxt] = False
                            current = nxt
                        else:
                            break
        for step in range(len(perm), city_count):
            d = dist[current].copy(); d[~unvisited] = np.inf
            valid = np.where(unvisited)[0]
            if len(valid) > 0:
                min_d = np.min(d[valid]); max_d = np.max(d[valid])
                range_d = max_d - min_d if max_d > min_d else 1.0
                noise = rng.random(city_count) * noise_scale * range_d
                if len(valid) > 2:
                    sub_dist = dist[np.ix_(valid, valid)];
                    avg_neigh = np.mean(sub_dist, axis=1)
                    avg_neigh_norm = (avg_neigh - np.min(avg_neigh)) /
                        (np.max(avg_neigh) - np.min(avg_neigh) + 1e-09)
                    noise[valid] -= 0.25 * avg_neigh_norm * range_d
                d += noise
        next_city = int(np.argmin(d)); perm.append(next_city)
        unvisited[next_city] = False; current = next_city
        new_candidates[i] = perm
        return new_candidates
```

Figure 6: Executable code of $O634$ in the best architecture obtained during training on Bi-TSP.

Bi-TSP | O634 ⟶ **O392**

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
    (batch_size, city_count) = candidates.shape;
    sel_idx =reference.selected_indices
    parent_objs = reference.parent_objectives[sel_idx];
    rank_sum = parent_objs.sum(axis=1)
    min_rs = rank_sum.min(); max_rs = rank_sum.max()
    if max_rs > min_rs:
        norm = (rank_sum - min_rs) / (max_rs - min_rs)
    else:
        norm = np.zeros_like(rank_sum)
    diff = 1.0 - norm; shuffle_prob = 0.5 + 0.2 * diff
    reverse_prob = 0.2 + 0.1 * diff; swap_prob = 0.3 - 0.1 * diff
    total = shuffle_prob + reverse_prob + swap_prob;
    shuffle_prob = shuffle_prob / total
    reverse_prob = reverse_prob / total; swap_prob = swap_prob / total
    max_block = min(10, city_count // 2); min_block = 2
    block_sizes = min_block + np.floor(diff*(max_block -min_block)).astype(int)
    block_sizes = np.clip(block_sizes, min_block, max_block);
    result = candidates.copy()
    rand_ops = rng.random(size=batch_size)
    starts = rng.integers(0, city_count, size=batch_size)
    directions = rng.integers(0, 2, size=batch_size)
    for i in range(batch_size):
        b_len = block_sizes[i]
        if b_len >= city_count:
            b_len = city_count - 1
        start = starts[i]; end = (start + b_len) % city_count
        if end > start:
            seg_indices = np.arange(start, end)
        else:
            seg_indices = np.arange(start, start + b_len) % city_count
        row = result[i]; seg_vals = row[seg_indices].copy()
        r = rand_ops[i]
        if r < shuffle_prob[i]:
            for j in range(len(seg_vals) - 1, 0, -1):
                k = int(rng.integers(0, j + 1))
                if k != j:
                    (seg_vals[j], seg_vals[k]) = (seg_vals[k], seg_vals[j])
            row[seg_indices] = seg_vals
        elif r < shuffle_prob[i] + reverse_prob[i]:
            row[seg_indices] = seg_vals[::-1]
        else:
            idx1 = int(rng.integers(0, city_count));
            idx2 = int(rng.integers(0,city_count))
            attempts = 0
            while idx2 == idx1 and attempts < 5:
                  idx2 = int(rng.integers(0, city_count)); attempts += 1
            if idx2 != idx1:
                (row[idx1], row[idx2]) = (row[idx2], row[idx1])
    return new_candidates
```

Figure 7: Executable code of $O392$ in the best architecture obtained during training on Bi-TSP.

Bi-KP | **O782** → O181 → O103 → O181

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
    (batch_size, item_count) = candidates.shape
    if reference.mating_pool is not None and reference.mating_pool.size > 0:
        pool = reference.mating_pool; pool_size = pool.shape[0]
        bias = pool.mean(axis=0);
        idx1 = rng.integers(0, pool_size, size=batch_size)
        idx2 = rng.integers(0, pool_size, size=batch_size); same = idx1 == idx2
        if np.any(same):
            idx2[same] = (idx2[same] + 1) % pool_size
            donor = pool[idx1]; target = pool[idx2]
        diff = np.logical_and(donor == 1, target == 0).astype(np.int8)
        scale = 0.5 * np.exp(-generation / 25.0) + 0.1
        apply_diff = (rng.random((batch_size, item_count)) < scale) & (diff==1)
        new_candidates = candidates.copy(); new_candidates[apply_diff] = 1
    else:
        flip_prob = 0.3 * np.exp(-generation / 40.0) + 0.05
        flip = rng.random((batch_size, item_count)) < flip_prob
    new_candidates = candidates.copy();
    new_candidates[flip] = 1 - new_candidates[flip]
    base_rate = 0.6 * np.exp(-generation / 20.0) + 0.05
    flip_prob = np.where(new_candidates == 0, base_rate * (1.0 + 0.5 *
        bias), base_rate * (0.5 - 0.3 * bias))
    flip_prob = np.clip(flip_prob, 0.0, 1.0)
    flip = rng.random((batch_size, item_count)) < flip_prob
    new_candidates[flip] = 1 - new_candidates[flip]
    if reference.mating_pool is not None and reference.mating_pool.size > 0:
        cross_prob = 0.2 * np.exp(-generation / 30.0) + 0.05
        do_cross = rng.random(batch_size) < cross_prob
        if np.any(do_cross):
            pool_size = reference.mating_pool.shape[0]
            src_idx = rng.integers(0, pool_size, size=batch_size);
            selected = np.where(do_cross)[0]
            cross_mask = rng.random((len(selected), item_count)) < 0.5
            new_candidates[selected] = np.where(cross_mask,
                reference.mating_pool[src_idx[selected]], new_candidates[selected])
    return new_candidates
```

Figure 8: Executable code of $O782$ in the best architecture obtained during training on Bi-KP.

Bi-KP | O782 → **O181** → O103 → O181

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
    (batch_size, item_count) = candidates.shape
    if reference.mating_pool is not None and reference.mating_pool.size>0:
        bias = reference.mating_pool.mean(axis=0)
    else:
        bias = 0.5 * np.ones(item_count)
    mask_prob = 0.9 * np.exp(-generation / 30.0) + 0.1
    mask = rng.random((batch_size, item_count)) < mask_prob
    flip_prob = np.where(candidates == 0,0.25*(1 + 2.0 * bias),0.25*(1 - 2.0*bias))
    flip_prob = np.clip(flip_prob, 0.0, 1.0)
    flip = (rng.random((batch_size, item_count)) < flip_prob) & mask
    new_candidates = candidates.copy();new_candidates[flip] = 1 - new_candidates[flip]
    if reference.mating_pool is not None and reference.mating_pool.size>0:
        inject_prob = 0.2 * np.exp(-generation / 50.0)
        do_inject = rng.random(batch_size) < inject_prob
        if np.any(do_inject):
            pool_size = reference.mating_pool.shape[0]
            source_indices = rng.integers(0, pool_size, size=batch_size)
            for i in range(batch_size):
                if do_inject[i]:
                    parent = reference.mating_pool[source_indices[i]]
                    seg_len = rng.integers(1, item_count // 2 + 1)
                    start = rng.integers(0, item_count - seg_len + 1)
                    new_candidates[i, start:start + seg_len] = parent[start:start + seg_len]
    return new_candidates
```

Bi-KP | O782 → O181 → **O103** → O181

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
    (batch_size, item_count) = candidates.shape
    if reference.mating_pool is not None and reference.mating_pool.size>0:
        bias = reference.mating_pool.mean(axis=0)
    else:
        bias = 0.5 * np.ones(item_count)
    noise_scale = max(0.0, 0.3 * (1.0 - generation / 100.0))
    noise = rng.normal(0.0, noise_scale, size=(batch_size, item_count))
    bias_matrix = np.tile(bias, (batch_size, 1)) + noise
    flip_prob = np.where(candidates == 0, 0.3 + 0.4 * (1.0 / (1.0 +
        np.exp(-bias_matrix))), 0.3 - 0.4 * (1.0 / (1.0 + np.exp(-bias_matrix))))
    flip_prob = np.clip(flip_prob, 0.01, 0.99)
    flip = rng.random((batch_size, item_count)) < flip_prob
    new_candidates = candidates.copy(); new_candidates[flip] = 1 - new_candidates[flip]
    if reference.parent_population is not None and reference.parent_population.size > 0:
        if reference.selected_indices is not None and len(reference.selected_indices) > 0:
            best_idx = reference.selected_indices[0]
        else:
            best_idx = 0
    best = reference.parent_population[best_idx]; new_candidates[0] = best.copy()
    return new_candidates
```

Figure 9: Executable code of $O181$ and $O103$ in the best architecture obtained during training on Bi-KP.

Bi-GVRP | **O385** → 0491 → 0123 → 0201

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
    (batch_size, perm_len) = candidates.shape;
    customer_count = context.customer_count
    selected = reference.selected_indices
    if len(selected) == 0:
        selected = np.arange(batch_size)
    parent_rows = rng.integers(0, len(selected), size=batch_size)
    sampled_parent_indices = selected[parent_rows];
    dist1 = context.distance_matrix_1
    dist2 = context.distance_matrix_2; dist = 0.6 * dist1 + 0.4 * dist2
    new_candidates = candidates.copy(); temp = 1.0 / (1.0 + 0.15 * generation)
    for i in range(batch_size):
        row_vals = candidates[i].copy()
        customer_positions = np.where(row_vals < customer_count)[0]
        if len(customer_positions) < 3:
            continue
        seg_len = rng.integers(2, min(4, len(customer_positions) + 1))
        start_idx = rng.integers(0, len(customer_positions) - seg_len +1)
        seg_positions = customer_positions[start_idx:start_idx + seg_len]
        seg_values = row_vals[seg_positions];
        mask = np.ones(perm_len, dtype=bool)
        mask[seg_positions] = False; row_without = row_vals[mask]
        insert_scores = np.full(len(row_without), 1000000000.0)
        remaining_customer_mask = row_without < customer_count
        remaining_customer_indices = np.where(remaining_customer_mask)[0]
        if len(remaining_customer_indices) > 0:
            for j in remaining_customer_indices:
                avg_dist = 0.0
                for val in seg_values:
                    avg_dist += dist[val, row_without[j]]
                    avg_dist /= len(seg_values); insert_scores[j] =  avg_dist
                    weights = np.exp(-insert_scores * temp);
                    weights_sum = weights.sum()
            if weights_sum > 0:
                probs = weights / weights_sum;
                insert_pos = rng.choice(len(row_without), p=probs)
            else:
                insert_pos = rng.integers(0, len(row_without))
        else:
            insert_pos = rng.integers(0, len(row_without))
        row_without = np.insert(row_without, insert_pos, seg_values)
        new_candidates[i] = row_without
    return new_candidates
```

Figure 10: Executable code of $O385$ in the best architecture obtained during training on Bi-GVRP.

Bi-GVRP | O385 → **0491** → 0123 → 0201

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
(batch_size, perm_len) = candidates.shape; customer_count = context.customer_count
    selected = reference.selected_indices
    if len(selected) == 0:
        selected = np.arange(batch_size)
    parent_rows = rng.integers(0, len(selected), size=batch_size)
    sampled_parent_indices = selected[parent_rows]; dist1 = context.distance_matrix_1
    dist2 = context.distance_matrix_2;
    blend = 0.5 + 0.3 * np.sin(generation / 10.0)
    dist = blend * dist1 + (1.0 - blend) * dist2; new_candidates = candidates.copy()
    temp = 1.0 / (1.0 + 0.05 * generation)
    for i in range(batch_size):
        row_vals = candidates[i].copy()
        customer_positions = np.where(row_vals < customer_count)[0]
        if len(customer_positions) < 2:
            continue
        if len(customer_positions) == 2:
            (pos1, pos2) = customer_positions
        else:
            (pos1, pos2) = rng.choice(customer_positions, size=2, replace=False)
        val1 = row_vals[pos1]; val2 = row_vals[pos2]
        row_without = np.delete(row_vals, [pos1, pos2])
        dist_from_val1 = np.full(perm_len, 1000000000.0)
        mask1 = (row_without < customer_count) & (row_without != val1)&(row_without != val2)
        valid_idx1 = np.where(mask1)[0]
        if len(valid_idx1) > 0:
            dist_from_val1[valid_idx1] = dist[val1, row_without[valid_idx1]]
        dist_from_val2 = np.full(perm_len, 1000000000.0)
        mask2 = (row_without < customer_count) & (row_without != val1)&(row_without != val2)
        valid_idx2 = np.where(mask2)[0]
        if len(valid_idx2) > 0:
            dist_from_val2[valid_idx2] = dist[val2, row_without[valid_idx2]]
        combined_dist = dist_from_val1 + dist_from_val2;
         weights = np.exp(-combined_dist * temp)
        weights_sum = weights.sum()
        if weights_sum > 0:
            probs = weights / weights_sum; new_pos1 = rng.choice(perm_len, p=probs)
            probs2 = probs.copy(); probs2[new_pos1] = 0.0
            if probs2.sum() > 0:
                probs2 = probs2 / probs2.sum(); new_pos2 = rng.choice(perm_len, p=probs2)
            else:
                new_pos2 = (new_pos1 + 1) % perm_len
        else:
            new_pos1 = rng.integers(0, perm_len); new_pos2 = rng.integers(0, perm_len - 1)
            if new_pos2 >= new_pos1:
                new_pos2 += 1
        if new_pos1 > len(row_without):
            new_pos1 = len(row_without)
        row_with_insert = np.insert(row_without, new_pos1, val1)
        if new_pos2 > len(row_with_insert):
            new_pos2 = len(row_with_insert)
        row_final = np.insert(row_with_insert, new_pos2, val2);
        new_candidates[i] = row_final
    return new_candidates
```

Figure 11: Executable code of $O491$ in the best architecture obtained during training on Bi-GVRP.

Bi-GVRP | O385 → 0491 → **0123** → 0201

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
(batch_size, perm_len) = candidates.shape; customer_count = context.customer_count
    selected = reference.selected_indices
    if len(selected) == 0:
        selected = np.arange(batch_size)
    parent_rows = rng.integers(0, len(selected), size=batch_size)
    sampled_parent_indices = selected[parent_rows];
    dist =context.distance_matrix_1
    new_candidates = candidates.copy(); temp = 1.0 / (1.0 + 0.02 * generation)
    for i in range(batch_size):
        row = candidates[i].copy();
        customer_positions = np.where(row<customer_count)[0]
        if len(customer_positions) < 2:
            continue
        (pos1, pos2) = rng.choice(customer_positions, size=2, replace=False)
        if pos1 > pos2:
            (pos1, pos2) = (pos2, pos1)
        segment = row[pos1:pos2 + 1].copy();
        seg_customers = segment[segment <customer_count]
        if len(seg_customers) < 2:
            continue
        unique_customers = np.unique(seg_customers)
        local_dist = np.zeros((len(unique_customers), len(unique_customers)))
        for a in range(len(unique_customers)):
            for b in range(len(unique_customers)):
                local_dist[a, b] = dist[unique_customers[a], unique_customers[b]]
            start_idx = 0; order = [start_idx]
            remaining = list(range(len(unique_customers)));
            remaining.remove(start_idx)
        while remaining:
            last = order[-1]; nearest = min(remaining, key=lambda idx:
            local_dist[last, idx])
            order.append(nearest); remaining.remove(nearest)
            reordered_customers = unique_customers[order]; new_segment =
            segment.copy()
        cust_counter = 0
        for j in range(len(segment)):
            if segment[j] < customer_count:
                new_segment[j] = reordered_customers[cust_counter]; cust_counter += 1
                new_row = row.copy(); new_row[pos1:pos2 + 1] = new_segment
            if rng.random() < 0.1:
                (inv_start, inv_end)=rng.choice(customer_positions,size=2,replace=False)
            if inv_start > inv_end:
                (inv_start, inv_end) = (inv_end, inv_start)
                new_row[inv_start:inv_end + 1] = new_row[inv_start:inv_end + 1][::-1]
        new_candidates[i] = new_row
    return new_candidates
```

Figure 12: Executable code of $O123$ in the best architecture obtained during training on Bi-GVRP.

Bi-GVRP | O385 → 0491 → 0123 → **0201**

```python
import numpy as np
def module(candidates, reference, context, generation, rng):
(batch_size, perm_len) = candidates.shape; customer_count = context.customer_count
    selected = reference.selected_indices
    if len(selected) == 0:
        selected = np.arange(batch_size)
    parent_rows = rng.integers(0, len(selected), size=batch_size)
    sampled_parent_indices = selected[parent_rows];
    dist = context.distance_matrix_1
    new_candidates = candidates.copy()
    for i in range(batch_size):
    row = candidates[i].copy();customer_positions = np.where(row<customer_count)[0]
        if len(customer_positions) < 2:
            continue
        parent_row = reference.parent_population[sampled_parent_indices[i]]
        parent_customers = parent_row[parent_row < customer_count]
        if len(parent_customers) < 2:
            continue
        unique_parent = np.unique(parent_customers)
        start_cust = unique_parent[rng.integers(0, len(unique_parent))];
        order = [start_cust]
        remaining = list(unique_parent[unique_parent != start_cust])
        while remaining:
            last = order[-1];
            distances = np.array([dist[last, c] for c in remaining])
            nearest_idx = int(np.argmin(distances));
            order.append(remaining[nearest_idx])
            remaining.pop(nearest_idx)
        if len(order) > len(customer_positions):
            order = order[:len(customer_positions)]
            seg_len = len(order)
        if seg_len < 2:
            continue
        start_pos = rng.integers(0, len(customer_positions) - seg_len + 1)
        seg_positions = customer_positions[start_pos:start_pos + seg_len];
        new_row = row.copy()
        for (j, pos) in enumerate(seg_positions):
            new_row[pos] = order[j]
        if rng.random() < 0.05:
            (inv_start, inv_end) = rng.choice(seg_positions, size=2, replace=False)
            if inv_start > inv_end:
                (inv_start, inv_end) = (inv_end, inv_start)
        new_row[inv_start:inv_end + 1] = new_row[inv_start:inv_end + 1][::-1]
        new_candidates[i] = new_row
    return new_candidates
```

Figure 13: Executable code of $O201$ in the best architecture obtained during training on Bi-GVRP.

Table 23: Effect of module pool size on MOSAIC with module rewriting disabled for Bi-TSP. Higher HV and lower IGD indicate better performance. The best and second-best mean values are highlighted in bold and underlined, respectively. Results are averaged over five independent runs.

Instance	Metric	20	100	200	500	1000	2000
kroAB100	HV	**0.739 ± 0.019**	0.420 ± 0.230	0.431 ± 0.288	0.587 ± 0.065	0.720 ± 0.053	0.586 ± 0.014
	IGD	0.193 ± 0.011	0.394 ± 0.145	0.381 ± 0.221	0.246 ± 0.044	**0.185 ± 0.050**	0.229 ± 0.011
kroAC100	HV	**0.729 ± 0.016**	0.383 ± 0.260	0.417 ± 0.281	0.590 ± 0.077	0.706 ± 0.050	0.552 ± 0.013
	IGD	0.176 ± 0.010	0.411 ± 0.188	0.364 ± 0.219	0.234 ± 0.037	**0.159 ± 0.050**	0.230 ± 0.001
kroAD100	HV	**0.711 ± 0.008**	0.361 ± 0.256	0.377 ± 0.298	0.569 ± 0.090	0.675 ± 0.051	0.486 ± 0.008
	IGD	0.180 ± 0.011	0.436 ± 0.193	0.417 ± 0.251	0.239 ± 0.066	**0.175 ± 0.036**	0.280 ± 0.002
kroAE100	HV	**0.732 ± 0.007**	0.387 ± 0.259	0.413 ± 0.302	0.586 ± 0.063	0.722 ± 0.049	0.555 ± 0.012
	IGD	0.178 ± 0.008	0.420 ± 0.194	0.385 ± 0.234	0.238 ± 0.044	**0.155 ± 0.036**	0.235 ± 0.004
kroBC100	HV	**0.725 ± 0.013**	0.362 ± 0.262	0.401 ± 0.293	0.587 ± 0.052	0.702 ± 0.046	0.540 ± 0.017
	IGD	0.181 ± 0.007	0.440 ± 0.208	0.402 ± 0.248	0.230 ± 0.039	**0.163 ± 0.038**	0.240 ± 0.013
kroBD100	HV	0.701 ± 0.021	0.377 ± 0.245	0.400 ± 0.301	0.570 ± 0.060	**0.701 ± 0.066**	0.529 ± 0.010
	IGD	0.194 ± 0.013	0.419 ± 0.177	0.395 ± 0.258	0.239 ± 0.043	**0.163 ± 0.058**	0.248 ± 0.010
kroBE100	HV	**0.721 ± 0.011**	0.389 ± 0.237	0.424 ± 0.271	0.596 ± 0.086	0.695 ± 0.042	0.543 ± 0.012
	IGD	0.193 ± 0.009	0.409 ± 0.169	0.372 ± 0.204	0.229 ± 0.065	**0.180 ± 0.047**	0.248 ± 0.007
kroCD100	HV	**0.718 ± 0.022**	0.375 ± 0.256	0.401 ± 0.294	0.581 ± 0.051	0.701 ± 0.045	0.540 ± 0.008
	IGD	0.202 ± 0.007	0.435 ± 0.182	0.412 ± 0.233	0.241 ± 0.045	**0.190 ± 0.030**	0.259 ± 0.007
kroCE100	HV	**0.727 ± 0.020**	0.387 ± 0.253	0.399 ± 0.296	0.598 ± 0.064	0.710 ± 0.037	0.537 ± 0.009
	IGD	0.196 ± 0.012	0.429 ± 0.163	0.412 ± 0.236	0.238 ± 0.045	**0.187 ± 0.037**	0.260 ± 0.011
kroDE100	HV	**0.725 ± 0.018**	0.383 ± 0.254	0.405 ± 0.291	0.577 ± 0.067	0.706 ± 0.041	0.531 ± 0.009
	IGD	0.180 ± 0.014	0.405 ± 0.176	0.389 ± 0.242	0.241 ± 0.038	**0.158 ± 0.046**	0.243 ± 0.010
kroAB150	HV	**0.727 ± 0.010**	0.352 ± 0.299	0.384 ± 0.332	0.561 ± 0.067	0.720 ± 0.058	0.555 ± 0.010
	IGD	0.200 ± 0.010	0.483 ± 0.225	0.451 ± 0.279	0.263 ± 0.050	**0.181 ± 0.042**	0.250 ± 0.007
kroAB200	HV	**0.727 ± 0.016**	0.331 ± 0.317	0.360 ± 0.343	0.536 ± 0.059	0.716 ± 0.056	0.578 ± 0.002
	IGD	0.203 ± 0.008	0.510 ± 0.252	0.477 ± 0.295	0.277 ± 0.053	**0.170 ± 0.038**	0.231 ± 0.003
euclidAB300	HV	0.704 ± 0.014	0.300 ± 0.335	0.316 ± 0.375	0.485 ± 0.042	**0.709 ± 0.075**	0.543 ± 0.010
	IGD	0.260 ± 0.010	0.561 ± 0.250	0.542 ± 0.314	0.325 ± 0.041	**0.211 ± 0.057**	0.267 ± 0.006
euclidCD300	HV	0.699 ± 0.014	0.309 ± 0.334	0.323 ± 0.373	0.499 ± 0.031	**0.716 ± 0.067**	0.557 ± 0.003
	IGD	0.238 ± 0.005	0.551 ± 0.266	0.526 ± 0.325	0.302 ± 0.033	**0.185 ± 0.053**	0.243 ± 0.003
euclidEF300	HV	0.694 ± 0.016	0.286 ± 0.343	0.304 ± 0.380	0.481 ± 0.039	**0.700 ± 0.072**	0.539 ± 0.006
	IGD	0.252 ± 0.006	0.586 ± 0.282	0.565 ± 0.341	0.321 ± 0.042	**0.199 ± 0.050**	0.261 ± 0.001
mixedGG300	HV	0.701 ± 0.020	0.314 ± 0.364	0.356 ± 0.404	0.476 ± 0.062	**0.776 ± 0.078**	0.610 ± 0.005
	IGD	0.273 ± 0.008	0.596 ± 0.279	0.522 ± 0.360	0.368 ± 0.078	**0.161 ± 0.090**	0.239 ± 0.001
mixedHH300	HV	0.707 ± 0.020	0.318 ± 0.357	0.355 ± 0.394	0.476 ± 0.058	**0.764 ± 0.089**	0.611 ± 0.004
	IGD	0.278 ± 0.012	0.589 ± 0.271	0.522 ± 0.359	0.369 ± 0.091	**0.167 ± 0.105**	0.232 ± 0.004
mixedII300	HV	0.690 ± 0.025	0.315 ± 0.361	0.351 ± 0.401	0.460 ± 0.042	**0.760 ± 0.088**	0.607 ± 0.009
	IGD	0.268 ± 0.008	0.591 ± 0.279	0.523 ± 0.356	0.368 ± 0.061	**0.169 ± 0.092**	0.238 ± 0.004
randomAB300	HV	0.717 ± 0.026	0.344 ± 0.399	0.394 ± 0.390	0.510 ± 0.077	**0.799 ± 0.048**	0.595 ± 0.002
	IGD	0.220 ± 0.001	0.585 ± 0.324	0.505 ± 0.318	0.331 ± 0.080	**0.173 ± 0.047**	0.249 ± 0.003
randomCD300	HV	0.715 ± 0.024	0.345 ± 0.399	0.390 ± 0.386	0.513 ± 0.067	**0.801 ± 0.047**	0.591 ± 0.006
	IGD	0.229 ± 0.007	0.588 ± 0.316	0.504 ± 0.316	0.335 ± 0.072	**0.174 ± 0.044**	0.257 ± 0.002
randomEF300	HV	0.718 ± 0.033	0.350 ± 0.405	0.399 ± 0.392	0.500 ± 0.059	**0.803 ± 0.055**	0.599 ± 0.003
	IGD	0.232 ± 0.003	0.582 ± 0.310	0.499 ± 0.321	0.337 ± 0.064	**0.171 ± 0.056**	0.247 ± 0.004
euclidAB500	HV	0.721 ± 0.015	0.295 ± 0.371	0.318 ± 0.398	0.453 ± 0.051	**0.734 ± 0.078**	0.595 ± 0.006
	IGD	0.257 ± 0.005	0.605 ± 0.296	0.565 ± 0.349	0.346 ± 0.031	**0.196 ± 0.060**	0.233 ± 0.004
euclidCD500	HV	0.717 ± 0.018	0.288 ± 0.370	0.311 ± 0.404	0.440 ± 0.050	**0.729 ± 0.082**	0.597 ± 0.002
	IGD	0.255 ± 0.002	0.614 ± 0.307	0.575 ± 0.366	0.352 ± 0.035	**0.193 ± 0.063**	0.230 ± 0.001
euclidEF500	HV	0.711 ± 0.016	0.282 ± 0.369	0.306 ± 0.403	0.443 ± 0.055	**0.723 ± 0.081**	0.579 ± 0.004
	IGD	0.260 ± 0.003	0.621 ± 0.309	0.583 ± 0.373	0.347 ± 0.034	**0.190 ± 0.069**	0.233 ± 0.003
mixedGG500	HV	0.608 ± 0.223	0.311 ± 0.388	0.347 ± 0.426	0.442 ± 0.070	**0.785 ± 0.087**	0.634 ± 0.002
	IGD	0.329 ± 0.153	0.625 ± 0.312	0.551 ± 0.384	0.394 ± 0.084	**0.167 ± 0.097**	0.225 ± 0.003
mixedHH500	HV	0.610 ± 0.224	0.310 ± 0.398	0.342 ± 0.428	0.447 ± 0.071	**0.784 ± 0.089**	0.634 ± 0.001
	IGD	0.338 ± 0.142	0.635 ± 0.309	0.556 ± 0.387	0.388 ± 0.099	**0.169 ± 0.100**	0.221 ± 0.001
mixedII500	HV	0.719 ± 0.022	0.310 ± 0.400	0.345 ± 0.432	0.436 ± 0.065	**0.790 ± 0.089**	0.639 ± 0.005
	IGD	0.267 ± 0.005	0.630 ± 0.319	0.560 ± 0.391	0.398 ± 0.087	**0.164 ± 0.092**	0.224 ± 0.003
randomAB500	HV	0.617 ± 0.217	0.325 ± 0.440	0.361 ± 0.434	0.483 ± 0.033	**0.816 ± 0.058**	0.613 ± 0.006
	IGD	0.299 ± 0.175	0.657 ± 0.383	0.572 ± 0.381	0.358 ± 0.057	**0.166 ± 0.041**	0.246 ± 0.004
randomCD500	HV	0.610 ± 0.219	0.325 ± 0.441	0.362 ± 0.433	0.470 ± 0.061	**0.814 ± 0.070**	0.610 ± 0.003
	IGD	0.305 ± 0.172	0.649 ± 0.376	0.569 ± 0.378	0.376 ± 0.081	**0.166 ± 0.049**	0.245 ± 0.002
randomEF500	HV	0.631 ± 0.210	0.343 ± 0.438	0.375 ± 0.429	0.492 ± 0.045	**0.825 ± 0.062**	0.637 ± 0.004
	IGD	0.281 ± 0.158	0.626 ± 0.370	0.550 ± 0.365	0.358 ± 0.079	**0.156 ± 0.044**	0.227 ± 0.001

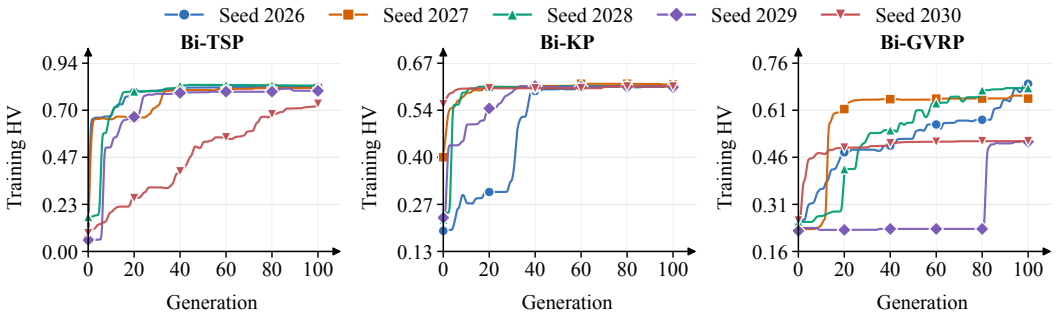


Figure 14: Training HV trajectories over five independent runs on Bi-TSP, Bi-KP, and Bi-GVRP under NSGA-II.

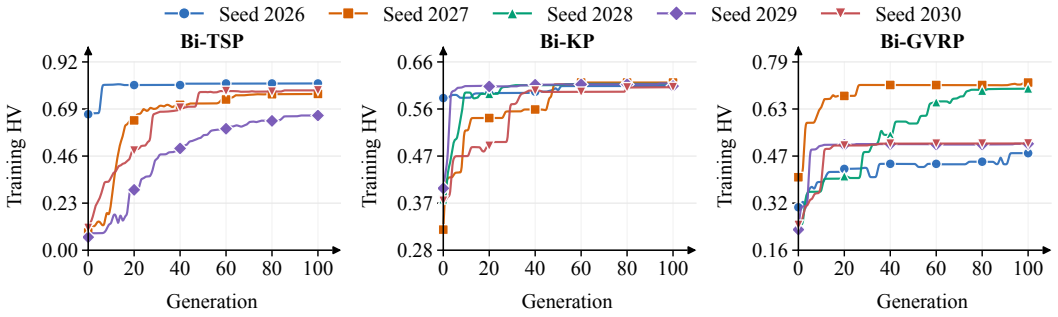


Figure 15: Training HV trajectories over five independent runs on Bi-TSP, Bi-KP, and Bi-GVRP under MOEA/D.

D.3 Additional Results on Initialization Sensitivity

This section provides additional results on the initialization sensitivity analysis in Section 4.3. Figs. 14 and 15 show the training HV trajectories over five independent runs on Bi-TSP, Bi-KP, and Bi-GVRP under the two multi-objective evolutionary algorithms (MOEAs), NSGA-II and MOEA/D, respectively. Under both MOEAs, architecture evolution generally improves architectures from different initial states, including those with relatively poor initial performance. Although some variation remains across runs, particularly on Bi-GVRP, the overall improvement from different initial states indicates that architecture evolution has limited sensitivity to initialization.

D.4 Additional Experiments on MOFJSP

To further examine the applicability of MOSAIC beyond the problem types and bi-objective settings considered in the main experiments, we conduct additional experiments on the multi-objective flexible job-shop scheduling problem (MOFJSP) using the Brandimarte MK01–MK15 instances (Brandimarte, 1993). We consider three minimization objectives: makespan $C_{\max}$, total workload $W_T = \sum_m W_m$, and maximum workload $W_M = \max_m W_m$.

A solution is represented by an integer chromosome of length $2|\mathcal{O}|$, encoding machine assignments and operation priorities. The decoder constructs a feasible schedule satisfying precedence, machine exclusivity, and non-preemption constraints. Accordingly, modules can modify either machine assignments or operation priorities, allowing MOSAIC to explore different scheduling behaviors through architecture evolution.

We use MK01–MK06, containing 10–15 jobs, for training and reserve MK07–MK15 for testing. MK07–MK10 contain 20 jobs, while MK11–MK15 contain 30 jobs, enabling evaluation on larger unseen instances. All other experimental settings are identical to those used in the main experiments.

Table 24: Per-instance test performance on MOFJSP. Higher HV indicates better performance. The best and second-best mean values are highlighted in bold and underlined, respectively. Results are reported as mean ± standard deviation over five independent runs.

Instance	Jobs	Machines	Ops.	Metric	EoH	FunSearch	ReEvo	MCTS-AHD	G-LNS	E2OC	MOSAIC
MK07	20	5	100	Mean	0.371	0.372	0.374	<u>0.375</u>	0.280	0.315	**0.378**
				Std.	0.004	0.006	0.008	0.005	0.018	0.008	0.025
MK08	20	10	225	Mean	0.170	0.173	0.173	<u>0.176</u>	0.145	0.151	**0.181**
				Std.	0.008	0.001	0.002	0.003	0.010	0.014	0.009
MK09	20	10	240	Mean	0.231	0.235	<u>0.237</u>	0.234	0.188	0.219	**0.244**
				Std.	0.004	0.003	0.004	0.005	0.014	0.009	0.013
MK10	20	15	240	Mean	0.284	0.286	0.285	<u>0.294</u>	0.231	0.267	**0.298**
				Std.	0.007	0.004	0.006	0.004	0.012	0.005	0.009
MK11	30	5	179	Mean	0.161	0.161	0.162	<u>0.163</u>	0.124	0.137	**0.164**
				Std.	0.005	0.003	0.003	0.002	0.008	0.008	0.005
MK12	30	10	193	Mean	0.178	0.176	<u>0.183</u>	0.179	0.148	0.155	**0.184**
				Std.	0.003	0.003	0.004	0.005	0.009	0.007	0.007
MK13	30	10	231	Mean	0.245	0.251	0.254	<u>0.257</u>	0.207	0.254	**0.266**
				Std.	0.001	0.006	0.002	0.005	0.011	0.010	0.011
MK14	30	15	277	Mean	0.163	0.164	0.165	<u>0.167</u>	0.136	0.160	**0.174**
				Std.	0.004	0.002	0.004	0.006	0.008	0.004	0.005
MK15	30	15	284	Mean	0.260	0.263	0.261	<u>0.269</u>	0.217	0.257	**0.272**
				Std.	0.005	0.006	0.003	0.005	0.010	0.009	0.010

Table 25: Architecture lengths across five independent runs with $L_{\max} = 4$.

Problem	NSGA-II	MOEA/D
Bi-TSP	3, 3, 2, 4, 4	4, 4, 4, 4, 4
Bi-KP	4, 4, 4, 4, 4	4, 4, 2, 4, 3
Bi-GVRP	4, 4, 4, 4, 4	4, 4, 4, 4, 4

As shown in Table 24, MOSAIC achieves the best mean HV on all nine test instances, including both the 20-job and 30-job groups. These results provide additional evidence for the applicability of MOSAIC to a distinct combinatorial optimization problem with three objectives, while also demonstrating generalization from smaller training instances to larger unseen instances.

D.5 Analysis of Architecture Length

Architecture length is evolved jointly with module composition and organization rather than fixed in advance, with $1 \leq L \leq L_{\max}$. Under the default setting $L_{\max} = 4$, Table 25 reports the lengths of the best architectures obtained during training across five independent runs.

Across the 30 runs, 25 architectures have length four, three have length three, and two have length two. The maximum length is selected in most runs, indicating that multi-stage architectures are frequently favored under the default search setting. Shorter architectures are also obtained in several runs, showing that the evolution process does not always favor the maximum allowed length.

To examine whether the frequent selection of $L = 4$ indicates that performance is constrained by the default upper bound, we further vary $L_{\max}$ over $\{2, 4, 6, 8\}$. All other experimental settings are identical to those used in the main experiments. As shown in Table 26, $L_{\max} = 4$ achieves the best overall performance across all three problems. Increasing the upper bound to 6 provides no further improvement, while $L_{\max} = 8$ leads to a clear performance decline. These results suggest that the frequent selection of $L = 4$ does not simply result from a restrictive upper bound. Instead, a larger architecture space may increase search difficulty under a fixed evaluation budget.

D.6 Limitations

Several aspects of the current framework warrant further investigation. First, existing LLM-driven AHD methods typically rely on prompting strategies or design ideas to generate heuristic code. However, whether the generated code faithfully implements the intended guidance and whether semantic drift occurs during generation remain difficult to systematically verify. MOSAIC also relies on prompt-guided code generation for module rewriting and therefore does not fully avoid this

Table 26: Sensitivity analysis of the maximum architecture length $L_{\max}$ under NSGA-II on Bi-TSP, Bi-KP, and Bi-GVRP. Best and second-best values are shown in bold and underlined, respectively. Results are averaged over five independent runs.

	Bi-TSP				Bi-KP				Bi-GVRP			
	Train	All Test			Train	All Test			Train	All Test		
$L_{\max}$	HV↑	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓	HV↑	HV↑	Gap↓	IGD↓
2	0.710	0.731	15.6%	0.139	0.563	0.560	33.7%	0.258	0.334	0.473	38.2%	0.271
4	**0.798**	**0.870**	**6.9%**	**0.069**	**0.610**	**0.591**	**29.8%**	**0.217**	**0.610**	**0.715**	**11.0%**	**0.091**
6	<u>0.774</u>	<u>0.847</u>	<u>9.6%</u>	<u>0.081</u>	<u>0.595</u>	<u>0.584</u>	<u>30.6%</u>	<u>0.218</u>	<u>0.603</u>	<u>0.710</u>	<u>11.2%</u>	<u>0.096</u>
8	0.742	0.797	14.6%	0.112	0.590	0.581	31.9%	0.220	0.403	0.562	27.4%	0.228

Table 27: Used assets and their licenses.

Type	Asset	License	Usage
Code	E2OC (Qiu et al., 2026)	MIT License	Evaluation
	G-LNS (Zhao et al., 2026)	MIT License	Evaluation
	EoH (Liu et al., 2024)	MIT License	Evaluation
	FunSearch (Romera-Paredes et al., 2024)	Apache-2.0 License	Evaluation
	ReEvo (Ye et al., 2024)	MIT License	Evaluation
	MCTS-AHD (Zheng et al., 2025)	MIT License	Evaluation
Dataset	Paquete Bi-objective TSP (Paquete & Stützle, 2003)	Public benchmark	Evaluation
	Xavier Bi-KP via Modolib.jl (Gandibleux & Freville, 2000)	Original-source terms	Evaluation
	Bi-GVRP (Erdoğdu & Karabulut, 2022)	Public benchmark	Evaluation
	Brandimarte FJSPLib (Brandimarte, 1993)	CC-BY-SA-4.0	Evaluation

issue. Our study with a large module pool shows that architecture evolution over fixed executable modules can already achieve promising performance. In particular, with a pool of 1,000 modules, its performance approaches that of existing LLM-driven AHD methods on Bi-TSP (see Appendix D.2). This suggests a possible direction for reducing the reliance of LLM-driven AHD on prompt-guided code generation.

Second, the efficiency of architecture evolution becomes increasingly important as the search space grows. Our experiments show that simply increasing the module pool size or the maximum architecture length does not consistently improve performance under a fixed evaluation budget. This motivates more efficient module selection and architecture search strategies to better exploit larger search spaces.

Finally, although the additional experiments on MOFJSP support the applicability of MOSAIC to a distinct combinatorial optimization problem with three objectives, the current evaluation covers a limited set of problem classes and settings. Further evaluation across additional problem classes, objective dimensions, and problem scales is needed to assess its broader applicability.

E LICENSES

We list the used assets and their licenses in Table 27.