LARGE LANGUAGE MODEL DRIVEN OPERATOR EVOLUTION: A PARADIGM FOR OPERATOR POOL RECONSTRUCTION AND ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Evolutionary search performance strongly depends on operator design, while most existing methods remain limited to predefined operators or search behaviors. This paper proposes a feedback-driven operator evolution framework that uses large language models (LLMs) to reconstruct search behaviors within a bounded parameter space. Candidate operators combine random exploration, constraint improvement, objective convergence, and diversity restoration, while offspring generation retains the original GA/DE variation interfaces. Candidate operators are evaluated according to their search performance, which provides feedback for LLM-based reconstruction. Reconstructed candidates are evaluated again before participating in main evolution, while the original operator remains available throughout the search. The framework is integrated into representative constrained multiobjective evolutionary algorithms. Experiments are conducted on the LIR-CMOP and DAS-CMOP benchmarks, together with the real-world problem sets RWMOP and MOOPF. Experimental results show improved overall performance across the tested problems.

1 INTRODUCTION

Evolutionary operators determine how candidate solutions are generated and therefore directly affect the exploration, convergence, and adaptability of evolutionary algorithms. Existing methods commonly improve search through strategy adaptation, parameter adaptation, or adaptive operator selection (Qin et al., 2009; Zhang & Sanderson, 2009; Li et al., 2014). Related studies have also shown that complementary search mechanisms can be beneficial when different optimization requirements dominate at different stages (Wang et al., 2019; Liu et al., 2022). However, the available variation behaviors are still largely predefined, and their effectiveness can change with the search state (Zou et al., 2024; Liu et al., 2025b). This motivates the automatic generation and evolution of search behaviors from online optimization feedback. This work investigates operator adaptation in constrained multiobjective optimization problems (CMOPs), where feasibility, convergence, and diversity impose changing search requirements during evolution (Liang et al., 2023). The formal definition and basic concepts of CMOPs are provided in Appendix A.

Large language models (LLMs) provide a new possibility for automated operator design because they can generate and modify algorithmic behaviors from structured feedback. The recent integration of LLMs with evolutionary computation has increasingly shifted from assisting optimization toward directly generating and evolving search behaviors (Wu et al., 2025; Liu et al., 2024b). Recent studies have combined LLMs with evolutionary search for program generation and automatic heuristic design (Romera-Paredes et al., 2024; Liu et al., 2024a), while execution-based feedback has also been used to iteratively improve generated algorithmic components (Ma et al., 2024). Reflective feedback and evolutionary variation have further been introduced into the generation process (Ye et al., 2024; Meyerson et al., 2024). Other work has explored code-level evolutionary search (Chen et al., 2023; Nasir et al., 2024), automatic metaheuristic generation (van Stein & Bäck, 2025), and broader exploration of LLM-generated algorithm spaces (Zheng et al., 2025; Hu & Zhang, 2025). LLMs have also been investigated as evolutionary search mechanisms themselves (Lange et al., 2024), and related ideas have been extended to multiobjective evolutionary optimization (Liu et al., 2025a). More recent studies extend this paradigm toward language-model and algorithm co-evolution and the

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.

evolution of complementary heuristic sets (Huang et al., 2026; Liu et al., 2026). Direct generation of executable evolutionary operators has also been investigated using online evolutionary feedback (Rao & Jia, 2026). However, executable-program generation tightly couples model outputs with the host optimizer, since generated operators must satisfy the decision encoding, variation interface, repair mechanism, and evaluation budget. This motivates a more structured and verifiable form of LLM-assisted operator evolution.

This work proposes a large language model feedback-driven operator evolution framework. Instead of synthesizing complete executable operators, the framework represents adaptable search behaviors through a bounded combination of random exploration, constraint improvement, objective convergence, and diversity restoration directions. Different directional weights and step sizes define different candidate behaviors while preserving the original GA/DE variation interfaces (Deb & Agrawal, 1995; Storn & Price, 1997). Candidate behaviors are evaluated through online screening using their actual search performance, and accumulated evidence is used to identify candidates that require reconstruction. The LLM modifies only bounded operator parameters. Validated candidates re-enter screening before becoming eligible for main evolution. The original operator is retained as a permanent anchor. In this way, the proposed framework enables feedback-driven operator evolution while limiting LLM intervention to a controlled and interpretable search space.

2 Related Work

2.1 Evolutionary Variation Mechanisms

Parameter definition. Let $P^t = \{\mathbf{x}_1^t, \ldots, \mathbf{x}_N^t\} \subset \mathbb{R}^D$ denote the population at generation t, where N is the population size and D is the number of decision variables. Each individual is represented by $\mathbf{x}_i^t = (x_{i,1}^t, \ldots, x_{i,D}^t)^{\mathrm{T}}$, and the jth decision variable is bounded by $l_j \le x_{i,j}^t \le u_j$.

Genetic Algorithm (GA). For real-valued optimization, GA generally generates offspring through crossover followed by mutation. Given two parent solutions $\mathbf{p}_1$ and $\mathbf{p}_2$, simulated binary crossover (SBX) produces $\mathbf{y}_1 = \frac{1}{2}[(1+\beta)\mathbf{p}_1 + (1-\beta)\mathbf{p}_2]$ and $\mathbf{y}_2 = \frac{1}{2}[(1-\beta)\mathbf{p}_1 + (1+\beta)\mathbf{p}_2]$, where the spread factor β is controlled by the crossover distribution index η_c (Deb & Agrawal, 1995). Polynomial mutation further perturbs each component as $y_j' = y_j + \Delta_j(u_j - l_j)$, where Δ_j is determined by the mutation distribution index η_m. Hence, the GA search behavior is mainly governed by parental recombination and the magnitude of subsequent perturbation.

Differential Evolution (DE). DE constructs offspring directly from differences among population members (Storn & Price, 1997). For the classical DE/rand/1 strategy, three mutually distinct individuals $\mathbf{x}_{r_1}^t$, $\mathbf{x}_{r_2}^t$, and $\mathbf{x}_{r_3}^t$ are sampled to generate the mutant vector $\mathbf{v}_i^t = \mathbf{x}_{r_1}^t + F(\mathbf{x}_{r_2}^t - \mathbf{x}_{r_3}^t)$, where $\mathbf{x}_{r_2}^t - \mathbf{x}_{r_3}^t$ defines the differential search direction and F controls its step size. The mutant vector is then recombined with the target vector $\mathbf{x}_i^t$ through binomial crossover. Specifically, $u_{i,j}^t = v_{i,j}^t$ if $\mathrm{rand}_j \le CR$ or $j = j_{\mathrm{rand}}$; otherwise, $u_{i,j}^t = x_{i,j}^t$, where CR denotes the crossover probability and j_{rand} ensures that at least one component is inherited from the mutant vector.

GA and DE provide complementary variation mechanisms with different forms of information recombination and directional search. These mechanisms can be further adapted or reconstructed through strategies, parameters, and additional search information.

2.2 Adaptive and Automatically Generated Operators

Strategy and parameter adaptation. Existing studies have extensively improved evolutionary operators through online strategy and parameter adaptation. SaDE maintains multiple DE mutation strategies and adjusts their selection probabilities and control parameters according to historical success (Qin et al., 2009). JADE further introduces the current-to-$p$best strategy, adaptive parameters, and an external archive to balance convergence and diversity (Zhang & Sanderson, 2009). Beyond DE-specific adaptation, adaptive operator selection evaluates the contributions of different candidate operators and dynamically allocates search opportunities according to their observed performance (Li et al., 2014). More recently, LAOS employs an LLM as a high-level controller to select operators according to population states and operator performance (Zhang & Yi, 2025). Although these methods improve adaptation to changing search conditions, the available strategies or operators are

still specified in advance, and online adaptation mainly determines how predefined behaviors are selected or configured.

Automatic operator generation. Instead of selecting among predefined operators, another line of research directly treats the operator structure as the object to be designed. Hong et al. used genetic programming as a generation hyper-heuristic to evolve mutation operators from probability distributions and arithmetic primitives, showing that operators can be specialized to individual function classes or generalized across multiple classes (Hong et al., 2016). Their later study introduced explicit training and testing over 23 function classes and demonstrated that automatically generated mutation operators can retain competitive performance on unseen instances (Hong et al., 2018). More recently, grammatical evolution has been used to automatically construct both crossover and mutation operators from low-level heuristic components for multiobjective optimization (Morales-Paredes et al., 2025). These approaches expand the operator space beyond manually enumerated alternatives, but the search space is still determined by predefined primitives or grammar rules, and generation is commonly performed through an offline outer evolutionary process.

LLM-assisted operator design. LLMs provide a more flexible way to generate and modify search behaviors. EoH and ReEvo use LLMs together with evolutionary feedback to generate and iteratively improve heuristic components (Liu et al., 2024a; Ye et al., 2024), while LLaMEA extends this paradigm to the automatic generation of complete metaheuristic algorithms (van Stein & Bäck, 2025). At the variation level, Language Model Crossover directly employs a language model as a crossover mechanism for text-representable genotypes (Meyerson et al., 2024). More closely related to this work, LLMAGEO uses online evolutionary feedback to trigger the generation of executable evolutionary operator code (Rao & Jia, 2026). These studies demonstrate that LLMs can extend variation beyond predefined operator structures. However, directly generating executable code also requires the generated operator to satisfy syntactic validity, decision encoding, variation interfaces, repair mechanisms, and runtime requirements.

The proposed framework combines online operator adaptation with LLM-based operator generation. Candidate behaviors are represented in a bounded parameter space and evaluated through online screening, while weak candidates are reconstructed by the LLM and continue to operate through the original GA/DE variation interfaces. This enables operator behaviors to evolve with the search state without relying on unrestricted executable-code generation.

3 Large Language Model Feedback Driven Operator Evolution Framework

Constrained multiobjective search requires different variation behaviors as feasibility, convergence, and diversity change in importance. Fixed operators may not accommodate these changes, whereas direct generation of executable operators by an LLM complicates validation. This work therefore restricts adaptation to operator selection and bounded parameter evolution.

The proposed framework adapts the variation process of the original CMOEA through an operator pool $\mathcal{A} = \{A_1, \ldots, A_K\}$, where A_1 is the original operator and the remaining members are parameterized candidates. Reproduction tasks are divided between main evolution and operator screening. The selected main operator A_m participates alongside A_1, while screening provides the performance evidence used for operator comparison and reconstruction. Offspring from both streams are returned to the original environmental selection, whereas only screening observations update operator rewards. When a mature candidate satisfies the reconstruction criterion, the LLM proposes revised parameters from the accumulated performance feedback. The validated update affects subsequent offspring generation without altering the population structure, environmental selection, or evaluation budget of the original CMOEA. Figure 1 summarizes the framework, while the detailed procedure is provided in Appendix B.1.

3.1 Parameterized Operator Generation Paradigm

The proposed paradigm parameterizes the search information supplied to the original variation operator rather than replacing its implementation. Four directions extracted from the reference population represent random exploration, constraint improvement, objective convergence, and diversity restoration. For each target solution $\mathbf{x}_i$, the random differential direction is defined as $\mathbf{d}_{i,R} = \mathbf{x}_{r_1} - \mathbf{x}_{r_2}$,

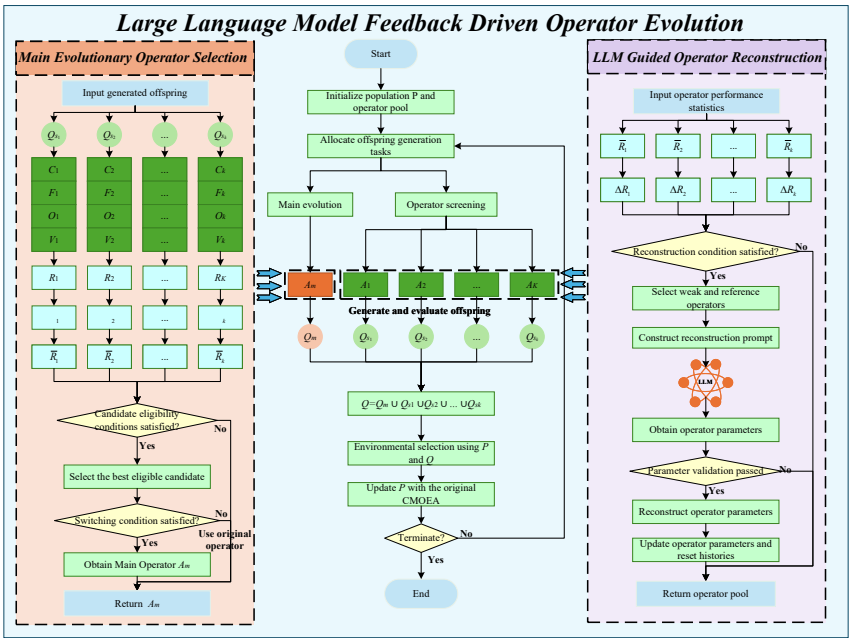


Figure 1: Overall framework

where $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$ are two differential reference solutions. The remaining three directions share the following form:

$$\mathbf{d}_{i,q} = \mathbf{x}_i^q - \mathbf{x}_i, \qquad q \in \{C, O, D\} \tag{1}$$

where $\mathbf{x}_i^C$, $\mathbf{x}_i^O$, and $\mathbf{x}_i^D$ provide constraint, objective, and diversity guidance, respectively. The four directions form a shared search basis rather than independent operators.

Each candidate A_k is parameterized by $\boldsymbol{\theta}_k = (w_{k,R}, w_{k,C}, w_{k,O}, w_{k,D}, \lambda_k)^\mathsf{T}$. The directional weights are nonnegative and normalized to sum to one, while $\lambda_k \in [0.1, 1]$ controls the search step. Different parameter vectors therefore represent different search preferences within the same bounded operator family. The corresponding auxiliary guide is

$$\mathbf{g}_{i,k} = \Pi_e \left(\mathbf{x}_i + \frac{\lambda_k}{\max\{1, (M-1)^2\}} \sum_{q \in \{R,C,O,D\}} w_{k,q} \mathbf{d}_{i,q} \right) \tag{2}$$

where M is the number of objectives and Π_e applies encoding-compatible repair. The repaired guide is supplied to the original GA or DE variation interface to generate offspring. The operator pool is initialized without invoking the LLM. Guide selection and interface details are provided in Appendix B.2.

3.2 Adaptive Operator Performance Evaluation

Operator rewards are estimated only from the screening stream. Given B variation units and screening budget S, the framework allocates $B_s = \min\{S, K, B\}$ units to screening and $B_m = B - B_s$ to main evolution. Screening opportunities are assigned to operators in rotating order to balance their accumulated observations. Before scoring, non-original offspring are compared with their paired parents through a deterministic protection rule. Rejected offspring are replaced by the corresponding parents after evaluation, while the consumed evaluations remain counted. Only the protected pairs assigned to the screening stream contribute to operator reward estimation, and the original operator is exempt from this rule.

For the n_k screening pairs $(\mathbf{x}_i, \mathbf{y}_{i,k})$ associated with A_k, let $\mathcal{U}_k$ contain pairs with an infeasible parent, $\mathcal{E}_k$ pairs with both solutions feasible, and $\mathcal{Y}_k$ pairs with a feasible offspring. Constraint progress is

measured by

$$C_k = \frac{1}{n_k} \sum_{i \in \mathcal{U}_k} \frac{[\widetilde{\mathrm{CV}}(\mathbf{x}_i) - \widetilde{\mathrm{CV}}(\mathbf{y}_{i,k})]_+}{\widetilde{\mathrm{CV}}(\mathbf{x}_i) + \epsilon}. \tag{3}$$

Here, $\widetilde{\mathrm{CV}}$ is the normalized aggregate constraint violation and $[\,\cdot\,]_+$ retains the positive part. Feasible-offspring generation is measured by

$$F_k = \frac{|\mathcal{Y}_k|}{n_k}. \tag{4}$$

For feasible parent–offspring pairs, objective improvement combines Pareto dominance with relative scalar improvement

$$O_k = \frac{1}{n_k} \sum_{i \in \mathcal{E}_k} \big[(1-\alpha) D_{i,k} + \alpha G_{i,k}\big], \tag{5}$$

where $D_{i,k}$ indicates whether the offspring dominates its parent, $G_{i,k} \in [0,1]$ measures relative scalar improvement, and $\alpha \in [0,1]$ balances the two terms. Diversity is measured by

$$V_k = \frac{1}{n_k} \sum_{i \in \mathcal{Y}_k} \frac{\min_{\mathbf{z} \in P} \angle(\widehat{\mathbf{F}}(\mathbf{y}_{i,k}), \widehat{\mathbf{F}}(\mathbf{z}))}{\pi}. \tag{6}$$

Here, $\widehat{\mathbf{F}}$ denotes the normalized objective vector. Detailed scalarization and normalization procedures are provided in Appendix B.3.

The four components are aggregated according to the feasible ratio FR of the reference population

$$R_k = (1 - \mathrm{FR})(0.7 C_k + 0.3 F_k) + \mathrm{FR}\big(\omega_O O_k + (1-\omega_O) V_k\big). \tag{7}$$

Here, $\omega_O \in [0,1]$ balances convergence and diversity. Low feasibility emphasizes constraint progress and feasible-offspring generation, whereas high feasibility shifts the reward toward objective improvement and diversity.

Each operator retains the latest H screening records $\mathbf{h}_k = (C_k, F_k, O_k, V_k, \mathrm{FR}, R_k)$, and its recent performance is represented by the mean reward $\overline{R}_k = |\mathcal{H}_k|^{-1} \sum_{\mathbf{h} \in \mathcal{H}_k} [\mathbf{h}]_R$. The original operator A_1 initially controls main evolution. A candidate is selected as A_m only when both it and A_1 have at least H records, its mean reward is the highest among eligible operators, and it exceeds $\overline{R}_1$ by the switching margin μ_{sw}. Otherwise, $A_m = A_1$. The selected candidate receives a bounded main-evolution share $\rho = \min\{\rho_{\max}, \max\{0, (\overline{R}_m - \overline{R}_1 - \mu_{\mathrm{sw}} - \beta_{\mathrm{tr}} u)/s\}\}$, where u is the standard error of its recent rewards, β_{tr} controls the uncertainty penalty, $\rho_{\max}$ limits candidate participation, and s normalizes the reward advantage. This limits candidate participation when its observed advantage is uncertain while retaining the original operator throughout the search.

3.3 Large Language Model Guided Operator Reconstruction

Reconstruction is checked every T_{rec} reproduction rounds, using the counter t. Let $\mathcal{V}$ index operators with nonempty finite reward histories. Their performance deficits are

$$\Delta R_k = \max_{j \in \mathcal{V}} \overline{R}_j - \overline{R}_k \tag{8}$$

Candidates with sufficient screening evidence form $\mathcal{M} = \{k \neq 1 : |\mathcal{H}_k| \geq H\}$, and $\mathcal{B} = \{k \in \mathcal{M} : \Delta R_k > \delta\}$ contains those eligible for replacement. The weakest eligible candidate has index k^-, and the operator with the highest observed mean reward has index k^+. When the optional first-reconstruction exception is enabled and $\mathcal{B}$ is empty, the worst candidate may be selected once all non-original candidates have H records. Each operator pool undergoes at most one reconstruction per scheduled check, subject to a cap on successful reconstructions.

The reconstruction context $\mathcal{Z}_{k^-,k^+}$ contains the problem dimensions, evaluation progress, operator family, and parameter and performance summaries of the weak and reference operators. The LLM returns a structured proposal containing four directional weights and a step size, not executable code. After validation and weight normalization, the proposal $\widehat{\boldsymbol{\theta}}_k$ is incorporated by

$$\boldsymbol{\theta}_k^{\mathrm{new}} = (1-\gamma)\boldsymbol{\theta}_k^{\mathrm{old}} + \gamma \widehat{\boldsymbol{\theta}}_k \tag{9}$$

where $\gamma \in [0, 1]$ controls interpolation between the previous parameters $\boldsymbol{\theta}_k^{\text{old}}$ and the proposal. The directional weights are renormalized, and the reconstructed candidate's reward and component histories are cleared before screening resumes. An invalid proposal leaves the current operator unchanged unless strict mode terminates execution. The complete prompt, validation rules, and reconstruction procedure accompany Algorithm 4 in Appendix B.4.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

All experiments are conducted on the PlatEMO platform with a population size of $N = 100$, a maximum of $100{,}000$ function evaluations, and 30 independent runs (Tian et al., 2017). The evaluation includes both benchmark CMOPs and practical engineering problems. LIR-CMOP is used to examine the ability of an algorithm to handle large infeasible regions and complex feasible-region structures, while DAS-CMOP contains different types and levels of constraint difficulty. To further evaluate practical applicability and generalization, 12 real-world constrained multiobjective problems from RWMOP and eight multiobjective optimal power flow (MOOPF) cases based on the IEEE 30-bus and IEEE 57-bus systems are considered. In total, the experiments include 14 LIR-CMOP, 9 DAS-CMOP, 12 RWMOP, and 8 MOOPF cases. IGD and HV are used for LIR-CMOP, DAS-CMOP, and RWMOP, while IGD^+ and HV are used for MOOPF. Their definitions and calculation details are provided in Appendix C.

Five representative CMOEAs, namely CCMO, C3M, C-TAEA, CMOEA-MS, and MOEA/D-DAE, are selected because they cover different constrained evolutionary mechanisms, including coevolutionary search, multi-stage search, complementary archives, stage-dependent search, and decomposition-based search with a detect-and-escape mechanism. Their enhanced versions are denoted as CCMO-LLM, C3M-LLM, C-TAEA-LLM, CMOEA-MS-LLM, and MOEA/D-DAE-LLM. All base algorithms adopt the default settings provided by PlatEMO. The proposed framework uses $\delta = 0.12$ and $\gamma = 0.75$. DeepSeek-V4-Flash is used as the default LLM, while MiMo-V2.5-Pro and Qwen-Plus are additionally considered in the cross-LLM experiments. Detailed framework parameters and LLM API settings are provided in Appendix D. Statistical significance is assessed using the two-sided Wilcoxon rank-sum test at the 0.05 level.

4.2 PERFORMANCE ON CONSTRAINED MULTIOBJECTIVE OPTIMIZATION PROBLEMS

4.2.1 STATISTICAL RESULTS AND ANALYSIS ON TEST PROBLEMS

Tables 1 and 2 present the HV and IGD results of CCMO and CCMO-LLM on the 35 test problems. The clearest improvements are observed on LIR-CMOP, particularly on LIR-CMOP5, LIR-CMOP6, LIR-CMOP8, LIR-CMOP9, LIR-CMOP10, and LIR-CMOP11. For example, on LIR-CMOP5, HV increases from 1.3858×10^{-1} to 2.8645×10^{-1}, while IGD decreases from 3.4074×10^{-1} to 1.5997×10^{-2}. Similar gains on LIR-CMOP6 and LIR-CMOP10 indicate that feedback-driven operator evolution can improve the search when the original variation behavior becomes ineffective. By screening operator performance and reconstructing weak candidates, the framework introduces additional search directions while retaining the original operator. On DAS-CMOP, the improvements are more problem dependent, with clear gains on DAS-CMOP1–3 and DAS-CMOP9 but degradation on DAS-CMOP5 and DAS-CMOP7. On RWMOP, HV improves on most problems, while several IGD results remain statistically similar. Overall, the framework provides larger benefits when additional search directions are needed, while retaining the original operator helps preserve stable performance. Complete results for the other four CMOEAs and their LLM-enhanced variants are provided in Appendix E. Convergence trajectories and final solution distributions on representative problems are reported in Appendix F.

4.2.2 COMPUTATIONAL COST ANALYSIS

Figure 2 reports the computational cost introduced by the LLM-assisted framework. The mean runtime of the original CMOEA-MS, CCMO, C-TAEA, C3M, and MOEA/D-DAE is 0.2, 0.4, 1.7, 0.6, and 1.1 minutes per independent run, respectively. After introducing the proposed framework, the corresponding total runtime increases to 1.7, 3.8, 4.5, 2.5, and 3.3 minutes. This additional cost

Table 1: HV results of CCMO and CCMO-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	1.1020e-1 (1.2318e-2)	2.2589e-1 (8.5978e-3)	9.5221e-2 (5.9251e-3)	1.8716e-1 (7.4740e-3)	1.3858e-1 (1.4203e-2)	1.0367e-1 (8.3483e-3)	2.3979e-1 (6.5920e-3)
LLM	**1.2089e-1** (2.5602e-2) =	**2.6804e-1** (3.0813e-2) +	**1.1531e-1** (2.1574e-2) +	**2.1039e-1** (2.2279e-2) +	**2.8645e-1** (1.6402e-3) +	**1.9266e-1** (1.1584e-3) +	**2.7994e-1** (1.3796e-2) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	2.2750e-1 (4.6552e-3)	3.1106e-1 (3.6975e-2)	5.9702e-1 (2.7808e-2)	6.4938e-1 (1.8299e-2)	4.7367e-1 (2.6641e-2)	**5.5339e-1** (1.5148e-3)	5.5321e-1 (1.0972e-3)
LLM	**2.8645e-1** (1.0200e-2) +	**5.2631e-1** (1.6216e-2) +	**6.9920e-1** (1.4018e-3) +	**6.9242e-1** (1.5389e-3) +	**6.1764e-1** (2.8927e-3) +	5.5109e-1 (1.7457e-3) −	**5.5499e-1** (1.7698e-3) +
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	4.4296e-3 (3.5351e-3)	2.5647e-1 (2.8570e-3)	2.0796e-1 (4.0044e-4)	1.9997e-1 (4.0205e-3)	**3.5129e-1** (1.0704e-4)	2.9504e-1 (1.7007e-2)	**2.8797e-1** (3.6910e-4)
LLM	**2.1020e-1** (1.2989e-3) +	**3.5423e-1** (2.7872e-4) +	**2.8561e-1** (1.8566e-2) +	**2.0242e-1** (3.2229e-3) +	3.5067e-1 (3.5730e-4) −	**3.0904e-1** (1.6580e-2) +	2.8697e-1 (6.6132e-4) −
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	**2.0685e-1** (3.9840e-4)	1.2072e-1 (8.2418e-3)	6.0697e-1 (2.8596e-4)	2.6760e-1 (1.6835e-1)	8.9884e-1 (5.6909e-4)	**8.5726e-1** (2.8468e-3)	4.3318e-1 (1.7105e-3)
LLM	2.0593e-1 (7.7497e-4) −	**2.0613e-1** (5.9746e-4) +	**6.0815e-1** (3.8184e-4) +	**3.9292e-1** (3.5356e-5) +	**8.9936e-1** (6.8281e-4) +	8.5681e-1 (4.9594e-3) =	**4.3504e-1** (9.1976e-5) +
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	**2.7696e-1** (4.4370e-5)	9.4245e-2 (9.7580e-4)	5.5239e-1 (4.4530e-3)	**8.7580e-2** (1.1861e-4)	5.4231e-1 (8.8112e-4)	3.9076e-1 (2.4964e-2)	4.0517e-2 (2.2906e-6)
LLM	2.7692e-1 (4.9292e-5) −	**9.4805e-2** (9.0169e-4) +	**5.5490e-1** (3.0208e-3) +	8.7556e-2 (1.2987e-4) =	**5.4295e-1** (3.3156e-4) +	**4.0139e-1** (1.2455e-2) +	**4.0519e-2** (2.1610e-6) +

Table 2: IGD results of CCMO and CCMO-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	3.0660e-1 (4.0020e-2)	2.7095e-1 (1.8474e-2)	3.2589e-1 (2.5386e-2)	3.0550e-1 (1.6768e-2)	3.4074e-1 (3.7148e-2)	3.8579e-1 (8.0698e-2)	1.4980e-1 (2.4992e-2)
LLM	**2.8194e-1** (6.9005e-2) =	**1.8274e-1** (6.4831e-2) +	**2.6271e-1** (6.9013e-2) +	**2.4840e-1** (4.9665e-2) +	**1.5997e-2** (3.2821e-3) +	**1.6406e-2** (4.1392e-3) +	**3.7079e-2** (3.1190e-2) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	2.1797e-1 (3.1030e-2)	6.0554e-1 (9.5957e-2)	1.9913e-1 (4.7523e-2)	9.0304e-2 (2.4816e-2)	3.0996e-1 (7.4262e-2)	**9.4126e-2** (7.1082e-4)	9.6742e-2 (7.4246e-4)
LLM	**2.2539e-2** (2.1602e-2) +	**1.3742e-1** (7.2505e-2) +	**1.5925e-2** (2.1167e-3) +	**4.7010e-3** (1.9670e-3) +	**8.6482e-3** (5.6115e-3) +	9.4711e-2 (1.1555e-3) −	**9.5728e-2** (8.9277e-4) +
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	7.4203e-1 (1.8669e-2)	2.5051e-1 (1.8675e-2)	3.5291e-1 (2.5161e-2)	2.6298e-3 (1.4220e-3)	**2.9133e-3** (8.9967e-5)	5.5410e-2 (4.6219e-2)	**3.0905e-2** (5.2701e-4)
LLM	**7.1524e-3** (2.5694e-3) +	**5.9469e-3** (5.2674e-4) +	**8.0377e-2** (5.6089e-2) +	**1.8415e-3** (1.1488e-3) =	3.2574e-3 (3.1004e-4) −	**2.9317e-2** (5.5164e-2) +	3.1435e-2 (1.0401e-3) −
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	**4.0300e-2** (7.7331e-4)	4.0833e-1 (4.8124e-2)	**3.5966e+5** (2.5587e+2)	**6.9511e+1** (2.9688e+1)	**4.3402e+1** (3.6459e+1)	**1.3605e+0** (1.4202e-2)	1.8882e+0 (4.3744e-5)
LLM	4.0698e-2 (2.9209e-3) =	**4.3584e-2** (3.0644e-3) +	3.5991e+5 (7.8618e+2) =	9.1241e+1 (1.0379e+0) −	8.0179e+1 (8.2330e+1) =	1.3644e+0 (9.5220e-3) =	**1.8882e+0** (9.8944e-5) +
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	6.0550e+2 (1.4955e-1)	2.4108e+6 (3.9179e+4)	2.8362e+0 (1.0022e+0)	3.8578e+2 (1.4939e+1)	1.2834e+3 (1.7175e+3)	1.5531e+11 (5.9896e+11)	9.4541e-2 (4.3379e-5)
LLM	**6.0545e+2** (3.0818e-2) +	**2.3668e+6** (5.1476e+4) +	**1.6751e+0** (9.8571e-1) +	**3.7742e+2** (1.2034e+1) +	**3.0988e+2** (7.9780e+2) +	**3.7470e+6** (9.2265e+6) =	**9.4514e-2** (6.6062e-5) =

mainly comes from LLM-related computation and API communication. The mean token consumption is 19.9, 40.1, 47.1, 29.8, and 32.3 thousand tokens per run, respectively. The corresponding mean numbers of LLM invocations are 41.4, 83.4, 98.1, 62.0, and 67.2. CMOEA-MS-LLM therefore introduces the lowest LLM overhead, while C-TAEA-LLM requires the highest token consumption and invocation frequency. Overall, the framework increases computational cost because of LLM interaction, but the number of calls remains limited compared with the overall evolutionary process. This is consistent with the event-driven reconstruction strategy, where the LLM is invoked only when operator reconstruction is required.

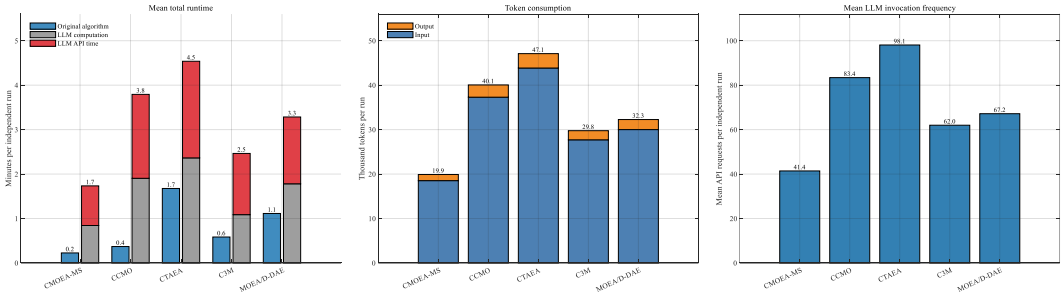


Figure 2: Computational costs of LLM-enhanced CMOEAs

4.2.3 Parameter Sensitivity Analysis

This study examines the sensitivity of δ and γ using CMOEA-MS-LLM on DAS-CMOP1–9. For each problem, Friedman ranks are calculated from 30 paired runs and then averaged over the nine problems (García et al., 2009). Table 3 shows that the default CMOEA-MS-LLM with ($\delta = 0.12, \gamma = 0.75$) achieves the lowest average ranks for both IGD and HV, with values of 3.6889 and 3.8833, respectively. For δ, the alternative settings generally produce higher average ranks, although CMOEA-MS-LLM$_{\delta=0.06}$ remains competitive. For γ, CMOEA-MS-LLM$_{\gamma=0.25}$ performs worst overall, while increasing γ to 0.50 improves the results. The default value $\gamma = 0.75$ achieves the best overall performance, whereas CMOEA-MS-LLM$_{\gamma=1.00}$ shows a slight degradation. The default configuration achieves the lowest average ranks among the tested settings. Detailed parameter settings and problem-level Friedman ranks are provided in Appendix G.

Table 3: Average Friedman ranks on DAS-CMOP.

Metric	CMOEA-MS-LLM	CMOEA-MS-LLM$_{\delta=0.06}$	CMOEA-MS-LLM$_{\delta=0.09}$	CMOEA-MS-LLM$_{\delta=0.15}$	CMOEA-MS-LLM$_{\delta=0.18}$	CMOEA-MS-LLM$_{\gamma=0.25}$	CMOEA-MS-LLM$_{\gamma=0.50}$	CMOEA-MS-LLM$_{\gamma=1.00}$
HV	**3.8833**	4.3870	4.5815	4.5815	4.7130	4.7315	4.5500	4.5722
IGD	**3.6889**	4.3741	4.6926	4.5519	4.7296	4.7259	4.5741	4.6630

4.3 Performance on Power Flow Problems

To further evaluate the practical applicability of the proposed framework, eight multiobjective optimal power flow (MOOPF) cases based on the IEEE 30-bus and IEEE 57-bus systems are considered. The cases involve different combinations of generation fuel cost, active power loss, pollutant emissions, and load voltage deviation, with two to four objectives. MATPOWER is used to perform AC power flow calculations for candidate solutions. The complete MOOPF formulation, control variables, objective functions, operating constraints, and test-case configurations are provided in Appendix H.

4.3.1 Performance with Different LLMs

Table 4 compares CMOEA-MS with CMOEA-MS-LLM using DeepSeek, MiMo, and Qwen. All three LLM variants achieve clear improvements on MOOPF2, MOOPF5, MOOPF6, and MOOPF8 for both IGD$^+$ and HV, indicating that the framework is not restricted to a specific LLM. The improvement is especially evident on MOOPF5, where all three variants substantially reduce IGD$^+$ and increase HV. MiMo shows the strongest overall performance among the three LLMs, although the differences are problem dependent. On MOOPF7, all three variants perform worse than the original CMOEA-MS, showing that LLM-guided operator evolution does not consistently benefit every power flow case.

Table 4: Performance of CMOEA-MS-LLM with different LLMs on the MOOPF problems.

Algorithm	Metric	MOOPF1	MOOPF2	MOOPF3	MOOPF4	MOOPF5	MOOPF6	MOOPF7	MOOPF8
CMOEA-MS	IGD$^+$	4.3949e-2 (3.74e-2)	1.7634e-3 (2.42e-4)	1.1202e-2 (9.54e-3)	2.4910e-2 (2.21e-3)	5.1447e-2 (2.30e-2)	5.1544e-3 (9.63e-4)	3.9270e-2 (1.50e-2)	7.6408e-2 (3.35e-2)
	HV	6.7461e-1 (9.31e-2)	8.3226e-1 (8.42e-4)	6.7898e-1 (3.96e-2)	6.4314e-1 (4.84e-3)	5.9187e-1 (9.71e-2)	7.9994e-1 (2.50e-3)	5.7812e-1 (3.87e-2)	4.4858e-1 (3.78e-2)
CMOEA-MS-LLM (DeepSeek)	IGD$^+$	7.3798e-2 (1.05e-1) −	**1.4997e-3** (9.98e-5) +	1.2165e-2 (8.71e-3) =	2.5478e-2 (1.83e-3) =	**1.5385e-2** (8.11e-3) +	**3.5806e-3** (1.29e-3) +	1.2555e-1 (8.14e-2) −	**4.7218e-2** (1.96e-2) +
	HV	6.7067e-1 (1.86e-1) −	**8.3393e-1** (8.45e-4) +	**6.7965e-1** (3.43e-2) =	**6.4686e-1** (3.22e-3) +	**8.1769e-1** (3.05e-2) +	**8.0446e-1** (4.01e-3) +	3.6699e-1 (1.12e-1) −	**5.0846e-1** (4.71e-2) +
CMOEA-MS-LLM (MiMo)	IGD$^+$	**3.7819e-2** (7.07e-2) +	**1.5001e-3** (5.89e-5) +	1.4815e-2 (1.21e-2) =	2.5852e-2 (2.17e-3) =	**1.4697e-2** (8.12e-3) +	**3.6712e-3** (1.22e-3) +	1.1794e-1 (5.73e-2) −	**5.9327e-2** (4.33e-2) +
	HV	**7.3066e-1** (1.34e-1) +	**8.3404e-1** (5.04e-4) +	6.7066e-1 (4.29e-2) =	**6.4616e-1** (3.84e-3) +	**8.2136e-1** (3.07e-2) +	**8.0443e-1** (3.57e-3) +	3.6026e-1 (9.59e-2) −	**4.8206e-1** (6.42e-2) +
CMOEA-MS-LLM (Qwen)	IGD$^+$	9.5527e-2 (1.12e-1) =	**1.5358e-3** (1.20e-4) +	1.6426e-2 (1.13e-2) −	2.5164e-2 (1.74e-3) =	**1.5644e-2** (8.28e-3) +	**3.4913e-3** (1.14e-3) +	1.1829e-1 (7.71e-2) −	**4.7677e-2** (2.32e-2) +
	HV	6.3033e-1 (1.96e-1) =	**8.3376e-1** (7.80e-4) +	6.6627e-1 (3.91e-2) =	**6.4730e-1** (3.13e-3) +	**8.1533e-1** (3.07e-2) +	**8.0464e-1** (3.52e-3) +	3.8320e-1 (1.38e-1) −	**5.0965e-1** (6.06e-2) +

Figure 3 further compares the final objective-space distributions on four representative biobjective cases. On MOOPF1 and MOOPF2, the four methods obtain largely overlapping tradeoff fronts. More evident differences appear on MOOPF5 and MOOPF6. The LLM-enhanced variants provide broader feasible coverage in the highlighted region of MOOPF5, while the enlarged view of MOOPF6 shows improved objective values at similar generation costs. These distributions are consistent with the corresponding IGD$^+$ and HV results and further demonstrate that the benefit of operator evolution varies across power flow cases.

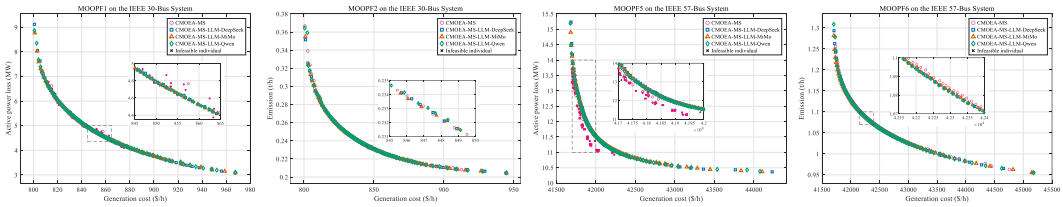


Figure 3: Objective-space solutions on biobjective MOOPF cases

4.3.2 Ablation Experiment Analysis

To evaluate the contributions of the main components, three ablation variants are constructed. CMOEA-MS-LLM-GD replaces the guided search directions with random differential directions, CMOEA-MS-LLM-AW replaces the adaptive performance weighting with fixed weights, and CMOEA-MS-LLM-LR replaces LLM-based reconstruction with parameter transfer from the current reference operator. Figure 4 compares their Friedman ranks with those of the complete CMOEA-MS-LLM on MOOPF1–8, where lower ranks indicate better performance. The complete framework obtains the lowest average ranks for both IGD^+ and HV, with values of 2.2458 and 2.1875, respectively. Replacing the guided directions produces the largest overall degradation, increasing the average ranks to 2.7042 and 2.6875. The AW and LR variants also yield higher average ranks than the complete framework, although their effects vary across individual problems. In particular, some ablation variants remain competitive on specific cases, indicating that no single component is uniformly beneficial on every problem. Overall, the complete framework achieves lower average ranks than the three ablation variants across the MOOPF cases. Complete problem-level results are reported in Appendix H.4.

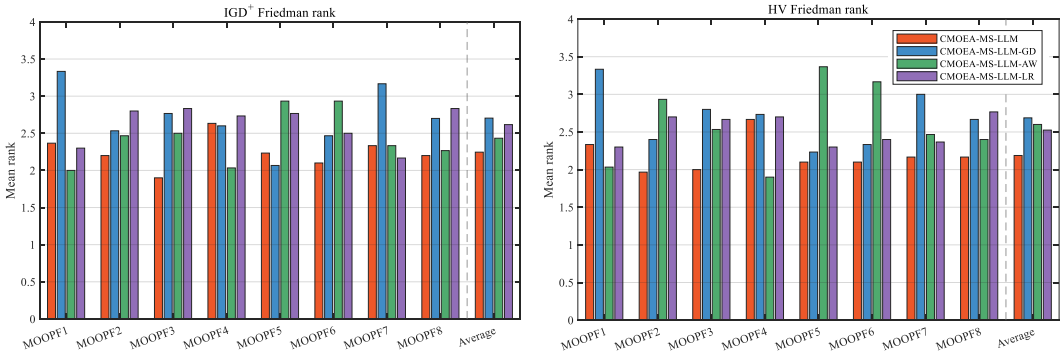


Figure 4: Friedman ranks of CMOEA-MS-LLM and its ablation variants on MOOPF.

5 Conclusion

This work proposes a feedback-driven operator evolution framework that uses LLMs to reconstruct evolutionary search behaviors based on online performance feedback. Candidate operators are represented in a bounded parameter space and generate offspring through the original GA/DE variation interfaces. Experimental results show that the framework improves overall performance across different original algorithms and problem settings, although the magnitude of improvement remains problem dependent. Ablation results support the contributions of guided search directions, adaptive performance weighting, and LLM-based reconstruction. Overall, the framework enables performance-triggered adaptation of operator parameters while preserving the original variation interfaces and environmental selection. Future work will investigate richer operator representations, more efficient reconstruction triggers, and lower LLM inference overhead, while extending the framework to other evolutionary algorithms and optimization problems.

AI USE STATEMENT

AI tools assisted manuscript writing and polishing, as well as the retrieval and discovery of related literature. The authors reviewed the AI-assisted content and take responsibility for the final paper.

REFERENCES

Angelica Chen, David Dohan, and David So. EvoPrompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, volume 36, pp. 7787–7817, 2023. doi: 10.52202/075280-0342. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/184c1e18d00d7752805324da48ad25be-Abstract-Conference.html.

Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995. URL https://www.complex-systems.com/abstracts/v09_i02_a02/.

Zhun Fan, Wenji Li, Xinye Cai, Hui Li, Caimin Wei, Qingfu Zhang, Kalyanmoy Deb, and Erik D. Goodman. Difficulty adjustable and scalable constrained multi-objective test problem toolkit. *Evolutionary Computation*, 28(3):339–378, 2020. doi: 10.1162/evco_a_00259. URL https://doi.org/10.1162/evco_a_00259.

Salvador García, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617–644, 2009. doi: 10.1007/s10732-008-9080-4. URL https://doi.org/10.1007/s10732-008-9080-4.

Libin Hong, John H. Drake, John R. Woodward, and Ender Özcan. Automatically designing more general mutation operators of evolutionary programming for groups of function classes using a hyper-heuristic. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 725–732. ACM, 2016. doi: 10.1145/2908812.2908958. URL https://doi.org/10.1145/2908812.2908958.

Libin Hong, John H. Drake, John R. Woodward, and Ender Özcan. A hyper-heuristic approach to automated generation of mutation operators for evolutionary programming. *Applied Soft Computing*, 62:162–175, 2018. doi: 10.1016/j.asoc.2017.10.002. URL https://doi.org/10.1016/j.asoc.2017.10.002.

Qinglong Hu and Qingfu Zhang. Partition to evolve: Niching-enhanced evolution with LLMs for automated algorithm discovery. In *Advances in Neural Information Processing Systems*, volume 38, pp. 171806–171852, 2025. doi: 10.52202/085713-5177. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e389b15166cf98966ba058965a8c17e3-Abstract-Conference.html.

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*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/75ed738c083f8941cfc990245e4f6265-Abstract-Conference.html.

Robert T. Lange, Yingtao Tian, and Yujin Tang. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 579–582. ACM, 2024. doi: 10.1145/3638530.3654238. URL https://doi.org/10.1145/3638530.3654238.

Ke Li, Álvaro Fialho, Sam Kwong, and Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014. doi: 10.1109/TEVC.2013.2239648. URL https://doi.org/10.1109/TEVC.2013.2239648.

Jing J. Liang, Xuanxuan Ban, Kunjie Yu, Boyang Qu, Kangjia Qiao, Caitong Yue, Ke Chen, and Kay Chen Tan. A survey on evolutionary constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 27(2):201–221, 2023. doi: 10.1109/TEVC.2022.3155533. URL https://doi.org/10.1109/TEVC.2022.3155533.

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 model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223. PMLR, 2024a. URL https://proceedings.mlr.press/v235/liu24bs.html.

Fei Liu, Xi Lin, Shunyu Yao, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Large language model for multiobjective evolutionary optimization. In *Evolutionary Multi-Criterion Optimization*, volume 15513 of *Lecture Notes in Computer Science*, pp. 178–191. Springer, 2025a. doi: 10.1007/978-981-96-3538-2_13. URL https://doi.org/10.1007/978-981-96-3538-2_13.

Fei Liu, Yilu Liu, Qingfu Zhang, Xialiang Tong, 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, 2026. doi: 10.1609/aaai.v40i43.41038. URL https://ojs.aaai.org/index.php/AAAI/article/view/41038.

Shaoning Liu, Jian Feng, Shengxiang Yang, Jun Zheng, and Qi Xiao. A coevolutionary algorithm with detection and supervision strategies for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 29(6):2343–2356, 2025b. doi: 10.1109/TEVC.2024.3416552. URL https://doi.org/10.1109/TEVC.2024.3416552.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation*, pp. 1–8. IEEE, 2024b. doi: 10.1109/CEC60901.2024.10611913. URL https://doi.org/10.1109/CEC60901.2024.10611913.

Zhi-Zhong Liu, Bing-Chuan Wang, and Ke Tang. Handling constrained multiobjective optimization problems via bidirectional coevolution. *IEEE Transactions on Cybernetics*, 52(10):10163–10176, 2022. doi: 10.1109/TCYB.2021.3056176. URL https://doi.org/10.1109/TCYB.2021.3056176.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IEduRUO55F.

Elliot Meyerson, Mark J. Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K. Hoover, and Joel Lehman. Language model crossover: Variation through few-shot prompting. *ACM Transactions on Evolutionary Learning and Optimization*, 4(4), 2024. doi: 10.1145/3694791. URL https://doi.org/10.1145/3694791.

Adrián Isaí Morales-Paredes, Julio Juárez, Jesús Guillermo Falcón-Cardona, Hugo Terashima-Marín, and Carlos A. Coello Coello. Automatic design of specialized variation operators for the multi-objective quadratic assignment problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1153–1161. ACM, 2025. doi: 10.1145/3712256.3726456. URL https://doi.org/10.1145/3712256.3726456.

Muhammad Umair Nasir, Sam Earle, Julian Togelius, Steven James, and Christopher Cleghorn. LLMatic: Neural architecture search via large language models and quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1110–1118. ACM, 2024. doi: 10.1145/3638529.3654017. URL https://doi.org/10.1145/3638529.3654017.

A. K. Qin, Vicky Ling Huang, and Ponnuthurai N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, 2009. doi: 10.1109/TEVC.2008.927706. URL https://doi.org/10.1109/TEVC.2008.927706.

Honghua Rao and Heming Jia. LLMAGEO: Large language model-assisted generation of evolutionary operators for constrained multiobjective optimization problems. *Applied Soft Computing*, 199: 115403, 2026. doi: 10.1016/j.asoc.2026.115403. URL https://doi.org/10.1016/j.asoc.2026.115403.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997. doi: 10.1023/A:1008202821328. URL https://doi.org/10.1023/A:1008202821328.

Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017. doi: 10.1109/MCI.2017.2742868. URL https://doi.org/10.1109/MCI.2017.2742868.

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. doi: 10.1109/TEVC.2024.3497793. URL https://doi.org/10.1109/TEVC.2024.3497793.

Jiahai Wang, Guanxi Liang, and Jun Zhang. Cooperative differential evolution framework for constrained multiobjective optimization. *IEEE Transactions on Cybernetics*, 49(6):2060–2072, 2019. doi: 10.1109/TCYB.2018.2819208. URL https://doi.org/10.1109/TCYB.2018.2819208.

Xingyu Wu, Sheng-Hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Transactions on Evolutionary Computation*, 29(2):534–554, 2025. doi: 10.1109/TEVC.2024.3506731. URL https://doi.org/10.1109/TEVC.2024.3506731.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1381. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4ced59d480e07d290b6f29fc8798f195-Abstract-Conference.html.

Jingqiao Zhang and Arthur C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009. doi: 10.1109/TEVC.2009.2014613. URL https://doi.org/10.1109/TEVC.2009.2014613.

Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. doi: 10.1109/TEVC.2007.892759. URL https://doi.org/10.1109/TEVC.2007.892759.

Yisong Zhang and Guoxing Yi. LAOS: Large language model-driven adaptive operator selection for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 517–526. ACM, 2025. doi: 10.1145/3712256.3726450. URL https://doi.org/10.1145/3712256.3726450.

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 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78338–78373. PMLR, 2025. URL https://proceedings.mlr.press/v267/zheng25o.html.

Junhua Zhu, Xiaobing Yu, Feng Wang, and Yaqi Mao. Multi-objective optimal power flow problem using constrained dynamic multitasking multi-objective optimization algorithm. *Swarm and Evolutionary Computation*, 93:101850, 2025. doi: 10.1016/j.swevo.2025.101850. URL https://doi.org/10.1016/j.swevo.2025.101850.

Juan Zou, Ruiqing Sun, Yuan Liu, Yaru Hu, Shengxiang Yang, Jinhua Zheng, and Ke Li. A multipopulation evolutionary algorithm using new cooperative mechanism for solving multiobjective problems with multiconstraint. *IEEE Transactions on Evolutionary Computation*, 28(1):267–280, 2024. doi: 10.1109/TEVC.2023.3260306. URL https://doi.org/10.1109/TEVC.2023.3260306.

A CMOP Definition and Basic Concepts

Constrained multiobjective optimization problems (CMOPs) involve multiple conflicting objectives subject to a set of constraints. Compared with unconstrained multiobjective optimization, CMOPs introduce an additional requirement on feasibility. Improving the objective values does not necessarily move a solution toward the feasible Pareto-optimal region. During evolution, the search process must therefore account for feasibility, convergence, and diversity simultaneously. The relative importance of these requirements may change with the problem structure and the current search state (Liang et al., 2023; Fan et al., 2020). Generally, a CMOP can be formulated as

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^{\mathrm{T}} \\ \text{s.t.} \quad & \begin{cases} g_j(\mathbf{x}) \le 0, & j = 1, \ldots, q, \\ h_j(\mathbf{x}) = 0, & j = q+1, \ldots, p, \\ l_i \le x_i \le u_i, & i = 1, \ldots, n. \end{cases} \end{aligned} \tag{10}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ is an n-dimensional decision vector and $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$ is the corresponding objective vector. $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$ denote the inequality and equality constraints, respectively. l_i and u_i are the lower and upper bounds of the ith decision variable. Since equality constraints are difficult to satisfy exactly in numerical optimization, a small positive tolerance δ is usually introduced. The violation of the jth constraint is then defined as

$$cv_j(\mathbf{x}) = \begin{cases} \max\{0, g_j(\mathbf{x})\}, & j = 1, \ldots, q, \\ \max\{0, |h_j(\mathbf{x})| - \delta\}, & j = q+1, \ldots, p. \end{cases} \tag{11}$$

The overall constraint violation is obtained by accumulating the violations of all constraints:

$$CV(\mathbf{x}) = \sum_{j=1}^{p} cv_j(\mathbf{x}). \tag{12}$$

The value of $CV(\mathbf{x})$ reflects the degree of infeasibility of a solution. A smaller value indicates that the solution is closer to satisfying all constraints. A solution is feasible when $CV(\mathbf{x}) = 0$; otherwise, it is infeasible. Accordingly, the feasible region can be expressed as

$$S = \{\mathbf{x} \in X \subseteq \mathbb{R}^n \mid CV(\mathbf{x}) = 0\}. \tag{13}$$

Constraints can also modify the Pareto-optimal structure of the original multiobjective problem. When the constraints are ignored, the nondominated front is referred to as the unconstrained Pareto front (UPF). After feasibility is taken into account, the feasible Pareto-optimal solutions form the constrained Pareto front (CPF). The relative positions of the CPF and UPF therefore reveal how strongly the constraints interfere with objective-oriented search. A close correspondence between the two fronts implies that objective improvement can also contribute to constrained convergence. As their discrepancy increases, however, search directions toward the UPF may become less useful for approaching the CPF and may even guide the population toward infeasible regions.

Figure 5 shows that the interaction between objective convergence and constraint satisfaction can vary considerably across CMOPs. In Fig. 5(a), the CPF coincides with the UPF, so objective improvement is consistent with convergence toward the constrained optimum. In Fig. 5(b), the two fronts remain largely consistent, although part of the UPF no longer corresponds to the feasible Pareto boundary. The discrepancy becomes more pronounced in Fig. 5(c), where the CPF and UPF only partially agree. In Fig. 5(d), the two fronts are clearly separated, and search directions toward the UPF may provide limited guidance for approaching the CPF. These relationships indicate that no single search preference is equally effective for all CMOPs or throughout the entire evolutionary process. Objective-oriented search is more useful when the CPF and UPF are highly consistent, whereas constraint improvement becomes increasingly important as the two fronts diverge. Once the population approaches the feasible Pareto region, diversity should also be maintained to obtain a well-distributed approximation of the CPF. Exploration remains useful when the current population provides insufficient search information or becomes trapped in a limited region. Therefore, random exploration, constraint improvement, objective convergence, and diversity restoration provide complementary search information, which motivates their adaptive combination in the proposed operator evolution framework (Liang et al., 2023; Liu et al., 2025b).

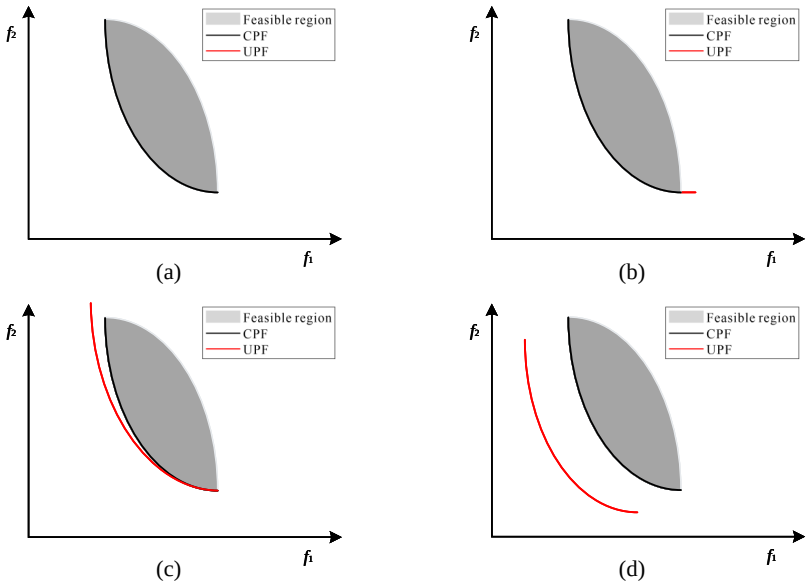


Figure 5: Representative CPF–UPF relationships in CMOPs.

B Algorithm Details

B.1 Overall Interface

Algorithm 1 presents the overall procedure. The method configuration is denoted by ψ and the total offspring budget by B^{tot}. The registered GA or DE interfaces are indexed by $G_\ell \in \mathcal{G} = \{G_1, \ldots, G_L\}$. Offspring Q_ℓ from the active channels are assembled into Q with repeated individuals retained and passed to the environmental selection of the original CMOEA.

Algorithm 1 Overall procedure of LLM feedback-driven operator evolution

Input: Original CMOEA $\mathcal{E}$, population size N, maximum number of function evaluations MaxFEs, and method configuration ψ
Output: Final population P

The original CMOEA initializes its populations and the framework initializes operator states
The initial evaluation count is assigned to FEs
while $\mathrm{FEs} < \mathrm{MaxFEs}$ **do**
    Determine B^{tot} without exceeding $\mathrm{MaxFEs} - \mathrm{FEs}$
    Allocate B^{tot} among the active variation channels
    for each active channel G_ℓ **do**
        Select channel-specific parents by tournament selection
        Generate offspring Q_ℓ and update the channel state using Algorithm 3
    end for
    Assemble Q from the channel offspring, retaining repeated individuals
    $\mathrm{FEs} \leftarrow \mathrm{FEs} + |Q|$
    The original CMOEA updates its populations using Q
    Allocation statistics are updated from environmental selection results
end while
return P

B.2 Initialization and Operator Interfaces

The candidate pool is initialized without invoking the LLM. Six predefined profiles provide different initial search preferences, including constraint improvement, objective convergence, diversity preservation, balanced search, joint constraint–objective search, and joint objective–diversity search. Small random perturbations are introduced to increase behavioral diversity, while additional candidates are generated within the same bounded parameter space. This initialization provides heterogeneous search behaviors for subsequent screening and reconstruction. During offspring generation, each candidate combines four sources of search information corresponding to random exploration, constraint improvement, objective convergence, and diversity restoration. The resulting search guidance is repaired according to the decision encoding and then passed to the original variation interface. For GA-based algorithms, it is used by the original full- or half-offspring GA operator, whereas for DE-based algorithms, it is incorporated through the original DE operator. Therefore, candidate operators modify only the search guidance supplied to variation, while the original crossover, mutation, solution evaluation, and environmental selection procedures remain unchanged.

Algorithm 2 summarizes the implementation of the parameterized operator generation procedure. It specifies how the differential references and three guide solutions are selected, how their directions are combined and repaired, and how the original variation operator generates the resulting offspring. The registered interface is denoted by $\tau \in \{\mathrm{GA}, \mathrm{GA}_{\mathrm{half}}, \mathrm{DE}\}$ and its implementation by $\mathcal{V}_0^\tau$. Guide repair follows the decision encoding e. Constraint guidance is drawn from the lowest-violation subset P^C, and objective guidance is drawn from the feasible first nondominated front F^1. Decision-space distances are normalized by variable bounds, whereas objective-space distances are normalized by the current componentwise objective ranges. These two normalizations are used only to select guides and do not modify objective or constraint values.

Algorithm 2 Offspring generation with an evolved operator

Input: Target $\mathbf{x}_i$, reference population P, selected parents, $\boldsymbol{\theta}_k$, M, e, τ, and $\mathcal{V}_0^\tau$
Output: Offspring $\mathcal{Q}_{i,k}$ generated for target $\mathbf{x}_i$

if $\tau = \mathrm{DE}$ **then**
    Select the differential references $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$ from the selected parents
else
    Sample distinct references $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$ from P
end if
Construct P^C from the lowest-violation 5% of P, retaining at least one solution
Select $\mathbf{x}_i^C$ as the nearest member of P^C with lower violation than $\mathbf{x}_i$, using P^C as the fallback set
Construct the feasible first nondominated front F^1
if $F^1 = \varnothing$ **then**
    $\mathbf{x}_i^O \leftarrow \mathbf{x}_i^C$
else
    Prefer feasible nondominated solutions that dominate $\mathbf{x}_i$, and otherwise retain the complete front as the guide set
    Select $\mathbf{x}_i^O$ as the nearest nonidentical objective guide, falling back to the first guide
end if
Select $\mathbf{x}_i^D$ as the most distant member of P in normalized decision space
Construct the search-direction basis from the selected references and guide solutions according to Eq. (1)
The repaired guide $\mathbf{g}_{i,k}$ is computed using Eq. (2)
Generate and evaluate $\mathcal{Q}_{i,k}$ through $\mathcal{V}_0^\tau$ using $\mathbf{x}_i$ and $\mathbf{g}_{i,k}$
return $\mathcal{Q}_{i,k}$

B.3 Screening Metrics and Acceptance

Before a non-original operator contributes either to the returned offspring or to its screening record, its raw offspring $\widetilde{Q}_k$ are checked against their paired parents. An offspring from an infeasible parent is admitted when its normalized constraint violation decreases. If both solutions are feasible, the offspring is admitted when it dominates the parent, reduces the value of a weighted Tchebycheff scalarizing function with weights derived from the parent, or provides sufficient angular novelty without exceeding the scalarization-loss tolerance. Otherwise, the paired parent is returned in its place. The resulting admitted set is denoted by Q_k. Because every raw offspring has already been evaluated, this protection changes neither the function-evaluation count nor the fixed offspring budget. The anchor operator is exempt from this filter.

Operator performance is estimated from the admitted parent–offspring pairs in the screening stream. To reduce the influence of different constraint scales, let q_j be the empirical 90th percentile of $|c_j(\mathbf{z})|$ over $\mathbf{z} \in P$. The normalized aggregate violation is

$$\widetilde{\mathrm{CV}}(\mathbf{x}) = \frac{1}{J}\sum_{j=1}^{J} \min\left\{1, \frac{\max\{0, c_j(\mathbf{x})\}}{\max\{q_j, \epsilon\}}\right\} \tag{14}$$

where J is the number of constraints and ϵ prevents division by zero. The clipping operation limits the contribution of an extreme violation.

Before computing convergence and diversity, each objective is shifted by the componentwise minimum of P, scaled by the empirical 90th percentile of the shifted values, and clipped to $[0, 1]$. This population-based normalization removes objective-scale effects without using the true Pareto front.

Convergence is evaluated only when both the parent and offspring are feasible. Tchebycheff scalarization combines multiple objectives into a scalar criterion and is widely used in decomposition approaches to multiobjective optimization (Zhang & Li, 2007). This work uses the following form with reciprocal weights derived from the parent

$$T(\mathbf{z} \mid \mathbf{a}_i) = \max_{1 \le j \le M} \frac{\widehat{f}_j(\mathbf{z})}{\max\{a_{i,j}, \epsilon\}} \tag{15}$$

where $\widehat{f}_j(\mathbf{z})$ is the normalized value of objective j and $a_{i,j} = (\widehat{f}_j(\mathbf{x}_i) + \epsilon)/\sum_{r=1}^{M}(\widehat{f}_r(\mathbf{x}_i) + \epsilon)$ is component j of the parent direction $\mathbf{a}_i$. The positive constant ϵ makes the direction well defined when the normalized parent vector is zero and prevents division by zero.

The same parent direction is used to evaluate both solutions. A lower value of T indicates improvement under this scalar criterion, including cases in which neither solution dominates the other. The relative scalar improvement is

$$G_{i,k} = \min\left\{1, \frac{[T(\mathbf{x}_i \mid \mathbf{a}_i) - T(\mathbf{y}_{i,k} \mid \mathbf{a}_i)]_+}{T(\mathbf{x}_i \mid \mathbf{a}_i) + \epsilon}\right\} \tag{16}$$

Algorithm 3 Operator screening and adaptive performance evaluation

Input: Current population P, selected parents, current channel state, and configuration $\boldsymbol{\psi}$
Output: Admitted offspring Q and updated channel state

The reproduction round t is incremented by one
Determine the available variation-unit count B
$B_s \leftarrow \min\{S, K, B\}$
$B_m \leftarrow B - B_s$
Sample the screening units uniformly and assign them to the operators in rotating order
Assign the remaining units to the anchor operator A_1
Retrieve the main-operator index m from the channel state and initialize $\rho \leftarrow 0$
if $m \neq 1$ and both A_1 and A_m have mature histories **then**
    The participation share ρ is computed from the reward advantage and uncertainty
end if
Reassign $\mathrm{round}(\rho B_m)$ randomly selected main-stream units to A_m
Initialize the offspring collection Q
for each active operator A_k **do**
    if $k = 1$ **then**
        Generate and evaluate Q_k using the original variation operator and its assigned parents
    else
        Generate $\widetilde{Q}_k$ for the assigned units using Algorithm 2
        Obtain Q_k by applying paired offspring protection to $\widetilde{Q}_k$
    end if
    Insert Q_k into its assigned positions in Q, retaining repeated individuals
    Retain the admitted pairs assigned to the screening stream
end for
for each operator represented in the screening stream **do**
    Normalize the constraint violation of each screened solution using Eq. (14)
    Evaluate the violation-reduction contribution C_k using Eq. (3)
    Evaluate the feasible-offspring contribution F_k using Eq. (4)
    Objective values are normalized using the reference population
    Evaluate the relative scalar improvement $G_{i,k}$ using Eq. (16)
    Evaluate the convergence contribution O_k using Eq. (5)
    Evaluate the diversity contribution V_k using Eq. (6)
    The feasible ratio FR is computed from P
    Aggregate the four contributions into the adaptive reward R_k using Eq. (7)
    Append $\mathbf{h}_k$ to $\mathcal{H}_k$ while retaining at most the latest H records
end for
The main operator is selected using mean rewards and the eligibility and switching criteria
Apply the performance-triggered reconstruction in Algorithm 4
Update the rotation cursor and the channel state
return Q and the updated channel state

B.4 LLM Prompt and Validation

The system prompt requests JSON output without executable code. The user prompt specifies the directional weights, step size, and adaptation instructions. The parser validates four finite nonnegative

weights with a sum above the numerical tolerance and a finite scalar step size in $[0.1, 1]$. Accepted weights are normalized before interpolation. The optional role descriptor defaults to `replacement`. Failed validation retains the current operator unless strict mode is enabled, in which case execution terminates with an error.

When the original operator A_1 provides the highest mean reward, predefined values fill its parameter fields in the prompt without changing its variation mechanism. Figure 6 presents the prompt and response interface for the adaptive scoring branch.

Prompt template used in the LLM reconstruction module
System prompt
You configure an evolutionary search operator. Return JSON only. Do not return MATLAB code, markdown, or explanations.
User prompt
The task has {*M*} objectives, {*J*} constraints, and {*D*} variables. Evaluation progress is {consumed evaluation fraction}. Replace a weak {operator family} operator. **Weak operator:** Previous role {role}, weights {weights}, step size {lambda}, and recent means {*C*, *F*, *O*, *V*, *FR*, *R*}. **Current best operator:** Role {best role}, weights {best weights}, step size {best lambda}, and recent means {best *C*, *F*, *O*, *V*, *FR*, *R*}. **Performance feedback:** *C* is robust per-constraint normalized violation improvement. *F* is the feasible-offspring ratio. *O* combines feasible Pareto dominance and direction-adaptive scalar convergence. *V* is normalized angular novelty. *FR* is the reference-population feasible ratio. The adaptive score is $R = (1 - FR)(0.7C + 0.3F) + FR(0.8O + 0.2V)$. When *FR* is low, favor the constraint direction. When *FR* is high, balance objective convergence and diversity. Use the performance gap and search progress to make a conservative change that can outperform the current best operator. Return one improved parameter set. The weights are ordered as [random, constraint, objective, diversity] and contain four nonnegative numbers summing to 1. The step size lambda is in [0.1, 1.0]. Return exactly the JSON structure below with your chosen numerical values.
LLM output
{"operator": { "role": "replacement", "weights": [number, number, number, number], "lambda": number }}

Figure 6: LLM prompt and response

Figure 7 and Figure 8 show the final saved operator parameters for the DE and GA interfaces on DAS-CMOP1, respectively, with $N = 100$ and 100,000 evaluations. These MATLAB representations describe stored parameter states rather than executable code generated by the LLM. Both pools retain the original PlatEMO operator. In the DE example, A_2 retains its initialized constraint-oriented profile, while A_3–A_9 are marked as LLM-updated. Their weights generally emphasize constraint improvement, with A_6 and A_7 assigning nearly equal weights to constraint and objective guidance. In the GA example, all eight candidates are marked as LLM-updated and assign their largest weight to objective guidance, ranging from 0.4234 to 0.4624. Most step sizes are close to 0.30, although GA candidate A_6 uses a smaller value of 0.2300 with a higher diversity weight of 0.2919. These single-run snapshots illustrate different parameter profiles under the two interfaces, but do not establish their relative effectiveness or consistency across runs.

<table>
<tr><th colspan="2">Original operator/DE</th></tr>
<tr><td colspan="2">% Original PlatEMO operator; not LLM-generated
operator = @OperatorDE;
% Direction weights and lambda: not applicable
% Source: Original platform operator
% Birth round: 0; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td></tr>
<tr><th>Operator 2</th><th>Operator 6</th></tr>
<tr><td>% Final saved parameters (MATLAB representation)
operator.role = 'constraint';
% Weights: random, constraint, objective, diversity
operator.weights = [0.10823406, 0.63359891, 0.14456496, 0.11360206];
operator.lambda = 0.27338361;
% Source: Initialization (recorded source: Random)
% Birth round: 0; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td><td>% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14873585, 0.34892082, 0.35130177, 0.15104155];
operator.lambda = 0.30225420;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 90; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td></tr>
<tr><th>Operator 3</th><th>Operator 7</th></tr>
<tr><td>% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.13832829, 0.45606703, 0.30409315, 0.10151153];
operator.lambda = 0.29685202;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 30; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td><td>% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.13706311, 0.36089792, 0.36182346, 0.14021550];
operator.lambda = 0.30169190;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 93; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td></tr>
<tr><th>Operator 4</th><th>Operator 8</th></tr>
<tr><td>% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14701949, 0.45148197, 0.29330728, 0.10819125];
operator.lambda = 0.29877571;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 102; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td><td>% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.13559606, 0.44171973, 0.31035950, 0.11232472];
operator.lambda = 0.29771649;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 75; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td></tr>
<tr><th>Operator 5</th><th>Operator 9</th></tr>
<tr><td>% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.15585623, 0.45652015, 0.24061387, 0.14700975];
operator.lambda = 0.29724642;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 66; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td><td>% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14849833, 0.44303199, 0.30248433, 0.10598536];
operator.lambda = 0.30113060;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 105; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run</td></tr>
</table>

Figure 7: Final DE operator parameters on DAS-CMOP1

Original operator/GA

```
% Original PlatEMO operator; not LLM-generated
operator = @OperatorGAhalf;
% Direction weights and lambda: not applicable
% Source: Original platform operator
% Birth round: 0; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 2

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14990240, 0.24842464, 0.45119528,
0.15047768];
operator.lambda = 0.30203583;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 93; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 6

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.10655167, 0.16887117, 0.43270927,
0.29186789];
operator.lambda = 0.22996163;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 27; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 3

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14027460, 0.20089013, 0.46241961,
0.19641566];
operator.lambda = 0.29197390;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 102; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 7

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14775646, 0.20343486, 0.44959051,
0.19921817];
operator.lambda = 0.29515857;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 54; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 4

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14938905, 0.19870551, 0.45479451,
0.19711094];
operator.lambda = 0.30808382;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 84; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 8

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.14955822, 0.20788865, 0.42340937,
0.21914376];
operator.lambda = 0.28835259;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 42; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 5

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.15040499, 0.19968725, 0.45034883,
0.19955892];
operator.lambda = 0.29202800;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 105; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Operator 9

```
% Final saved parameters (MATLAB representation)
operator.role = 'replacement';
% Weights: random, constraint, objective, diversity
operator.weights = [0.16381742, 0.21567129, 0.43996267,
0.18054863];
operator.lambda = 0.29126286;
% Source: LLM-updated (DeepSeek Falsh)
% Birth round: 33; final FE: 100000; N: 100
% DASCMOP1; full framework configuration; one run
```

Figure 8: Final GA operator parameters on DAS-CMOP1

Algorithm 4 Performance-triggered LLM operator reconstruction

Input: Operator pool $\mathcal{A}$, current search state, and configuration ψ
Output: Updated operator pool $\mathcal{A}$

Construct $\mathcal{V}$ from the indices of operators with nonempty finite reward histories
if $t \bmod T_{\mathrm{rec}} = 0$, the reconstruction budget remains, and $\mathcal{V} \neq \varnothing$ **then**
    Mean rewards are computed from the histories of operators in $\mathcal{V}$
    $k^{+} \leftarrow \arg\max_{k \in \mathcal{V}} \overline{R}_k$
    Performance deficits are computed using Eq. (8)
    $\mathcal{M} \leftarrow \{k \neq 1 \mid |\mathcal{H}_k| \geq H\}$
    $\mathcal{B} \leftarrow \{k \in \mathcal{M} \mid \Delta R_k > \delta\}$
    if $\mathcal{B} \neq \varnothing$ **then**
        $k^{-} \leftarrow \arg\min_{k \in \mathcal{B}} \overline{R}_k$
    else if cold-start replacement is enabled, no earlier reconstruction has succeeded, and all non-original candidates are mature **then**
        $k^{-} \leftarrow \arg\min_{k \in \mathcal{V} \setminus \{1\}} \overline{R}_k$
    else
        $k^{-} \leftarrow \varnothing$
    end if
    if $k^{-} \neq \varnothing$ **then**
        Assemble $\mathcal{Z}_{k^{-},k^{+}}$ from the search state and the weak and best operator summaries
        Query the LLM for a structured parameter proposal
        if the proposal passes deterministic parsing and validation **then**
            Normalize the proposed directional weights
            Compute the reconstructed parameter vector by the bounded interpolation in Eq. (9)
            Normalize the reconstructed directional weights
            Install the reconstructed role and parameter vector at index k^{-} in $\mathcal{A}$
            Clear the reward and component histories of the reconstructed operator before subsequent screening
            Update the successful-reconstruction count
        else if strict mode is enabled **then**
            Execution terminates with a validation error
        end if
    end if
end if
return $\mathcal{A}$

C PERFORMANCE EVALUATION INDICATORS

Let $\mathcal{A}$ denote the objective vectors of the final feasible nondominated solutions obtained by an algorithm, and let $\mathcal{R}$ denote the reference set used for performance evaluation. The inverted generational distance (IGD) is defined as

$$\mathrm{IGD}(\mathcal{A}, \mathcal{R}) = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \min_{\mathbf{a} \in \mathcal{A}} \|\mathbf{r} - \mathbf{a}\|_2. \tag{17}$$

IGD jointly reflects convergence and coverage of the obtained solution set. A smaller value indicates better performance.

For the MOOPF experiments, IGD^+ is used to reduce the influence of improvements in one objective compensating for deterioration in another. It is calculated as

$$\mathrm{IGD}^+(\mathcal{A}, \mathcal{R}) = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \min_{\mathbf{a} \in \mathcal{A}} d^+(\mathbf{r}, \mathbf{a}), \tag{18}$$

where

$$d^+(\mathbf{r}, \mathbf{a}) = \sqrt{\sum_{m=1}^{M} [\max(a_m - r_m, 0)]^2}. \tag{19}$$

A smaller IGD^+ value indicates a better approximation of the reference front.

The hypervolume (HV) measures the objective-space region dominated by the obtained nondominated set and bounded by a predefined reference point $\mathbf{r}^{\mathrm{HV}}$. For a minimization problem, it is defined as

$$\mathrm{HV}(\mathcal{A}) = \lambda_M \left(\bigcup_{\mathbf{a} \in \mathcal{A}} \prod_{m=1}^{M} [a_m, r_m^{\mathrm{HV}}] \right), \tag{20}$$

where $\lambda_M(\cdot)$ denotes the M-dimensional Lebesgue measure. A larger HV value indicates better convergence and diversity.

For each problem, the same reference information is used for all compared algorithms. Each indicator is calculated independently for every run, and the mean and standard deviation over 30 independent runs are reported. IGD and HV are used for LIR-CMOP, DAS-CMOP, and RWMOP, while IGD^+ and HV are used for MOOPF.

D Framework Parameter Settings

Unless otherwise stated, the same framework parameters are used for all CMOEAs. Each registered variation channel maintains an operator pool containing the original operator and eight parameterized candidates, with the original operator permanently retained as the anchor. Candidate initialization does not invoke the LLM. The key parameters controlling operator screening, selection, and reconstruction are kept fixed across all experiments to ensure consistent integration with different CMOEAs. Their settings are summarized in Table 5.

Table 5: Default parameter settings of the proposed framework.

Parameter	Symbol	Value
Operator pool size	K	9
Screening budget	S	9
History length	H	10
Reconstruction threshold	δ	0.12
Reconstruction interval	T_{rec}	3
Reconstruction interpolation	γ	0.75

Three LLMs from different providers are employed to evaluate the model generality of the proposed framework. DeepSeek-V4-Flash is used as the default LLM, while MiMo-V2.5-Pro and Qwen-Plus are additionally considered in the cross-LLM experiments. Since the LLM is only required to generate compact operator parameters, the thinking mode is disabled for all three models. The temperature, output-token budget, timeout, and maximum number of API attempts are kept identical across models. The detailed API settings are summarized in Table 6.

Table 6: LLM API settings used in the experiments.

Parameter	DeepSeek-V4-Flash	MiMo-V2.5-Pro	Qwen-Plus
API model	`deepseek-v4-flash`	`mimo-v2.5-pro`	`qwen-plus`
Provider	DeepSeek	Xiaomi MiMo	Alibaba Cloud DashScope
Context length	1M	1M	1M
Thinking mode	Disabled	Disabled	Disabled
Temperature	0	0	0
Top-p	API default	API default	API default
Max output tokens	1200	1200	1200
Stream	false	false	false
Timeout	120 s	120 s	120 s
Max API attempts	3	3	3

E Detailed Experimental Results

Tables 7, 8, 9, 10, 11, 12, 13, and 14 present the detailed results of the other four CMOEAs and show a similar overall pattern to that observed for CCMO. The most evident improvements again occur on LIR-CMOP, where the LLM-enhanced variants generally obtain higher HV and lower IGD values than their original versions. For C3M, the improvement is particularly consistent across this suite. On LIR-CMOP5, HV increases from 2.0591×10^{-1} to 2.8330×10^{-1}, while IGD decreases from 3.0910×10^{-1} to 2.1717×10^{-2}. C-TAEA exhibits an even larger change on LIR-CMOP8, where HV increases from 5.4603×10^{-2} to 2.5302×10^{-1} and IGD decreases from 1.2741 to 1.1927×10^{-1}. Similar improvements are observed for CMOEA-MS on LIR-CMOP5–12 and for MOEA/D-DAE on LIR-CMOP1–8. For example, on LIR-CMOP2, MOEA/D-DAE-LLM increases HV from 2.3457×10^{-1} to 3.4079×10^{-1} and reduces IGD from 2.0654×10^{-1} to 3.9565×10^{-2}. These results further indicate that the additional search directions generated through feedback-driven operator evolution are useful when the original variation behavior has difficulty crossing infeasible barriers or maintaining sufficient progress toward different feasible regions. In contrast, the results

on DAS-CMOP are more dependent on both the problem and the original CMOEA. C-TAEA-LLM achieves clear improvements on DAS-CMOP1–6, whereas C3M-LLM shows limited gains on several later DAS-CMOP problems, and MOEA/D-DAE-LLM deteriorates on cases such as DAS-CMOP4. The RWMOP results show a similar tendency. Clear improvements can still be observed on difficult cases such as RWMOP12, RWMOP13, RWMOP15, and RWMOP17 for several enhanced algorithms. For instance, CMOEA-MS-LLM reduces the IGD on RWMOP15 from 4.2865×10^3 to 2.4363×10^2, but a number of RWMOP cases remain statistically similar or occasionally become worse. This variation across algorithms and problems is expected because the proposed framework does not replace the original constraint-handling and environmental-selection mechanisms. Instead, it modifies reproduction according to online operator feedback. Therefore, the largest improvements occur when the original variation behavior lacks an effective search direction, while the benefit becomes smaller or may disappear when the original CMOEA already provides suitable search behavior for the current objective–constraint landscape.

Table 7: HV results of C3M and C3M-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	1.2415e-1 (1.2812e-2)	2.2222e-1 (2.8799e-2)	1.1051e-1 (2.1120e-2)	2.0706e-1 (2.9881e-2)	2.0591e-1 (1.0358e-1)	1.5001e-1 (5.4019e-2)	2.8423e-1 (1.8598e-2)
LLM	**1.7026e-1** (1.6080e-2) +	**2.7648e-1** (3.2891e-2) +	**1.3784e-1** (1.7846e-2) +	**2.3093e-1** (3.2376e-2) +	**2.8330e-1** (3.5798e-3) +	**1.8568e-1** (1.6590e-2) +	**2.9348e-1** (1.1175e-3) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	2.9043e-1 (1.0217e-2)	3.8600e-1 (1.8585e-2)	5.8555e-1 (4.0274e-2)	5.7974e-1 (6.0706e-2)	5.3166e-1 (2.1244e-2)	3.8974e-1 (1.6391e-1)	4.3498e-1 (1.6066e-1)
LLM	**2.9381e-1** (2.7258e-4) +	**4.7879e-1** (4.4468e-2) +	**6.5758e-1** (6.7462e-2) +	**6.6145e-1** (7.7072e-2) +	**5.7849e-1** (3.6727e-2) +	**5.2873e-1** (2.5617e-3) +	**5.4865e-1** (1.9461e-3) +
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	2.1037e-1 (1.3427e-3)	3.5446e-1 (1.3816e-4)	2.7641e-1 (2.3596e-2)	**1.6180e-1** (7.1454e-2)	**3.2703e-1** (6.4562e-2)	**2.8312e-1** (7.2524e-2)	**2.4073e-1** (4.6369e-2)
LLM	**2.1155e-1** (9.6259e-4) +	**3.5504e-1** (9.4492e-5) +	**2.8494e-1** (2.6445e-2) +	1.3358e-1 (7.1531e-2) =	2.6446e-1 (1.2341e-1) −	1.5994e-1 (1.2259e-1) =	1.9198e-1 (8.1128e-2) =
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	**1.6268e-1** (4.8245e-2)	**2.0364e-1** (6.7681e-4)	6.0802e-1 (3.1083e-4)	3.9286e-1 (1.7854e-5)	8.9911e-1 (4.8778e-4)	8.5722e-1 (1.6599e-3)	4.3456e-1 (1.0737e-4)
LLM	1.4788e-1 (6.1551e-2) =	2.0026e-1 (1.9683e-2) −	**6.0822e-1** (3.5908e-4) +	**3.9292e-1** (1.7700e-5) +	**8.9973e-1** (2.9342e-4) +	**8.5968e-1** (8.6409e-4) +	**4.3481e-1** (8.0655e-5) +
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	2.7605e-1 (1.9530e-4)	8.1870e-2 (4.3926e-3)	5.5592e-1 (1.6924e-3)	8.7054e-2 (2.4105e-4)	5.4359e-1 (1.8117e-5)	3.8869e-1 (1.8358e-2)	4.0512e-2 (2.0145e-6)
LLM	**2.7673e-1** (1.0128e-4) +	**9.0319e-2** (3.0565e-3) +	**5.5613e-1** (1.7361e-3) =	**8.7357e-2** (1.8050e-4) +	**5.4366e-1** (1.4049e-5) +	**3.9032e-1** (2.5807e-2) =	**4.0516e-2** (2.0961e-6) +

Table 8: IGD results of C3M and C3M-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	2.5287e-1 (4.1799e-2)	2.3814e-1 (5.5692e-2)	3.2673e-1 (1.2715e-1)	2.6393e-1 (6.6333e-2)	3.0910e-1 (4.1930e-1)	1.4726e-1 (1.8225e-1)	3.0975e-2 (4.8434e-2)
LLM	**1.6589e-1** (5.5697e-2) +	**1.5552e-1** (5.6969e-2) +	**1.7071e-1** (5.6377e-2) +	**2.0378e-1** (6.9152e-2) +	**2.1717e-2** (9.5329e-3) +	**4.4248e-2** (7.2293e-2) +	**8.7379e-3** (2.2861e-3) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	1.5498e-2 (2.6962e-2)	4.2925e-1 (5.2638e-2)	2.4615e-1 (6.6570e-2)	1.7955e-1 (8.9057e-2)	1.6971e-1 (3.7804e-2)	3.2131e-1 (3.6243e-1)	2.6753e-1 (3.1854e-1)
LLM	**8.2248e-3** (4.1230e-4) +	**2.4751e-1** (9.3316e-2) +	**1.0180e-1** (1.3158e-1) +	**5.5025e-2** (1.1640e-1) +	**8.7431e-2** (7.2868e-2) +	**1.0686e-1** (1.9795e-3) +	**9.8184e-2** (1.2510e-3) +
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	8.1727e-3 (5.2969e-3)	4.9933e-3 (1.3560e-4)	1.8544e-1 (8.3734e-2)	**1.4953e-1** (3.3165e-1)	**4.3706e-2** (1.2575e-1)	**7.9935e-2** (1.7565e-1)	**1.2997e-1** (1.3995e-1)
LLM	**4.4020e-3** (1.3645e-3) +	**4.4645e-3** (1.3283e-4) +	**1.1787e-1** (1.0360e-1) +	1.7724e-1 (1.8360e-1) =	1.5300e-1 (2.1579e-1) −	3.2049e-1 (2.4687e-1) =	2.4475e-1 (1.9703e-1) =
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	**1.5681e-1** (1.9233e-1)	**4.3583e-2** (2.0032e-3)	3.5990e+5 (8.3440e+2)	9.1220e+1 (7.3192e-1)	1.3192e+2 (1.2931e+2)	1.3695e+0 (1.6084e-3)	1.8882e+0 (5.1964e-5)
LLM	1.5901e-1 (1.3471e-1) =	6.5288e-2 (8.1747e-2) −	**3.5989e+5** (6.5431e+2) =	**9.0829e+1** (1.2817e+0) =	**4.7151e+1** (4.7828e+1) +	1.3695e+0 (1.7376e-3) =	**1.8881e+0** (2.2181e-4) =
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	**6.0536e+2** (5.8742e-2)	**2.2726e+6** (5.1926e+4)	2.7941e+0 (8.6696e-1)	4.0673e+2 (2.2869e+1)	1.2599e+2 (7.5367e+1)	3.0425e+5 (7.0029e+5)	9.4553e-2 (2.5588e-5)
LLM	6.0542e+2 (4.8211e-2) −	2.2828e+6 (6.4900e+4) =	**2.0777e+0** (1.2744e+0) +	**3.8322e+2** (1.5944e+1) +	**3.5961e+1** (1.4401e+1) +	**2.0271e+5** (2.4752e+5) =	**9.4517e-2** (9.8709e-5) =

Table 9: HV results of C-TAEA and C-TAEA-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	1.0192e-1 (1.6710e-2)	2.2200e-1 (3.7064e-2)	7.7911e-2 (1.1838e-2)	1.7504e-1 (1.1772e-2)	0.0000e+0 (0.0000e+0)	0.0000e+0 (0.0000e+0)	1.8799e-1 (7.6497e-2)
LLM	**1.3659e-1** (2.9788e-2) +	**2.5999e-1** (3.0141e-2) +	**1.1433e-1** (2.1134e-2) +	**2.0653e-1** (2.1442e-2) +	**2.9463e-2** (6.4813e-2) +	**6.4989e-3** (2.0073e-2) =	**2.5816e-1** (1.2517e-2) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	5.4603e-2 (8.8100e-2)	2.9018e-1 (1.1207e-3)	4.9394e-1 (2.9910e-2)	5.7302e-1 (2.3838e-2)	4.4084e-1 (1.9720e-2)	5.4558e-1 (1.0320e-3)	**5.4571e-1** (7.1467e-4)
LLM	**2.5302e-1** (1.7772e-2) +	**4.9381e-1** (1.9483e-2) +	**6.9313e-1** (1.9543e-3) +	**6.8867e-1** (2.4731e-3) +	**6.0173e-1** (5.7013e-3) +	**5.4700e-1** (1.1420e-3) +	5.4366e-1 (1.8118e-3) −
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	1.6059e-1 (1.5342e-2)	2.9650e-1 (4.9229e-3)	2.4043e-1 (1.1253e-2)	1.9003e-1 (2.3571e-3)	3.4714e-1 (6.3525e-4)	3.0589e-1 (3.2132e-3)	**2.8654e-1** (2.2244e-3)
LLM	**2.0884e-1** (1.6818e-3) +	**3.4755e-1** (1.0478e-2) +	**3.1041e-1** (1.3529e-3) +	**2.0025e-1** (3.9295e-3) +	**3.4945e-1** (5.6685e-4) +	**3.1183e-1** (3.2318e-4) +	2.8172e-1 (1.9954e-3) −
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	**2.0299e-1** (2.0430e-3)	1.3284e-1 (5.5673e-3)	**6.0600e-1** (7.9186e-4)	2.5500e-1 (1.8326e-1)	8.8158e-1 (9.5821e-3)	**8.4594e-1** (9.6831e-3)	4.3106e-1 (1.6404e-3)
LLM	1.9936e-1 (1.8843e-3) −	**1.9688e-1** (6.2775e-3) +	6.0554e-1 (8.5898e-4) −	**3.9295e-1** (5.4268e-5) +	**8.9874e-1** (1.4902e-3) +	8.3681e-1 (1.3841e-2) −	**4.3484e-1** (3.8805e-4) +
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	1.9987e-1 (5.7304e-2)	9.3791e-2 (8.6518e-4)	5.4112e-1 (8.2405e-3)	**8.6038e-2** (9.5823e-4)	5.3591e-1 (1.5916e-2)	2.3094e-1 (3.5418e-2)	4.0313e-2 (8.1219e-5)
LLM	**2.7621e-1** (1.1122e-3) +	**9.4393e-2** (1.0811e-3) +	**5.4886e-1** (7.7122e-3) +	8.5526e-2 (8.3036e-4) −	**5.4250e-1** (3.8355e-4) +	**2.6179e-1** (1.4953e-2) +	**4.0394e-2** (5.9423e-5) +

Table 10: IGD results of C-TAEA and C-TAEA-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	3.7326e-1 (6.2762e-2)	3.1352e-1 (1.1534e-1)	4.3412e-1 (9.9441e-2)	3.6186e-1 (5.7486e-2)	1.2330e+0 (6.5684e-3)	1.3463e+0 (4.7677e-4)	4.4668e-1 (4.8215e-1)
LLM	**2.3341e-1** (7.8148e-2) +	**1.9447e-1** (5.4413e-2) +	**2.5629e-1** (6.6166e-2) +	**2.5594e-1** (4.8380e-2) +	**1.0627e+0** (3.2491e-1) +	**1.2646e+0** (2.5359e-1) +	**9.1558e-2** (3.1774e-2) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	1.2741e+0 (6.0570e-1)	5.9902e-1 (3.2029e-2)	3.9481e-1 (4.4602e-2)	2.8092e-1 (5.2541e-2)	3.7501e-1 (1.0345e-1)	1.1035e-1 (1.7211e-3)	**1.1154e-1** (1.0066e-3)
LLM	**1.1927e-1** (6.4099e-2) +	**2.1814e-1** (4.8367e-2) +	**2.4859e-2** (3.0671e-3) +	**9.2369e-3** (3.3050e-3) +	**4.1518e-2** (1.1236e-2) +	**1.0885e-1** (1.9774e-3) +	1.1254e-1 (2.5752e-3) =
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	2.0779e-1 (3.7243e-2)	1.2085e-1 (2.0210e-2)	1.8554e-1 (5.4180e-2)	1.3993e-2 (1.3603e-3)	8.4762e-3 (5.4961e-4)	2.9449e-2 (4.6913e-3)	**4.0448e-2** (6.2645e-3)
LLM	**8.6225e-3** (2.5439e-3) +	**1.7571e-2** (2.7146e-2) +	**2.1098e-2** (2.1938e-3) +	**3.7083e-3** (1.5450e-3) +	**5.3599e-3** (5.8910e-4) +	**1.9899e-2** (1.2119e-3) +	4.6163e-2 (4.1927e-3) −
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	**5.6586e-2** (6.3912e-3)	3.1256e-1 (3.6275e-2)	1.7945e+6 (6.8535e+5)	**8.1254e+1** (2.9454e+1)	4.4686e+4 (1.7124e+4)	1.3404e+0 (5.2246e-2)	1.8965e+0 (2.0545e-2)
LLM	6.1467e-2 (5.5034e-3) −	**7.4608e-2** (1.4193e-2) +	**1.0706e+6** (6.0322e+5) +	9.0922e+1 (1.8704e+0) −	**2.3430e+4** (1.1882e+4) +	**1.3028e+0** (2.4029e-1) =	**1.8917e+0** (1.3336e-2) =
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	**2.3751e+3** (6.4125e+2)	2.5547e+6 (1.5688e+4)	2.4433e+0 (8.5969e-1)	2.1097e+3 (5.1445e+2)	2.7962e+3 (4.1248e+3)	5.5718e+3 (1.3593e+3)	9.4547e-2 (4.1664e-5)
LLM	2.6997e+3 (4.7545e+2) =	**2.5329e+6** (3.3467e+4) +	**1.4812e+0** (1.0904e+0) +	**9.0336e+2** (1.9801e+2) +	**1.4992e+3** (2.7127e+3) +	**4.8243e+3** (2.0204e+2) +	**9.4454e-2** (2.3093e-4) =

Table 11: HV results of CMOEA-MS and CMOEA-MS-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	9.7226e-2 (7.2787e-3)	2.1189e-1 (1.3605e-2)	9.1786e-2 (4.2713e-3)	1.7609e-1 (1.1756e-2)	1.1026e-1 (5.2291e-2)	7.5989e-2 (3.8916e-2)	2.3997e-1 (5.5741e-3)
LLM	**1.0816e-1** (1.4036e-2) +	**2.2838e-1** (1.7452e-2) +	**9.6054e-2** (7.7060e-3) +	**1.8901e-1** (1.2671e-2) +	**2.0734e-1** (1.2748e-1) +	**1.2413e-1** (6.5122e-2) +	**2.5176e-1** (1.0558e-2) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	2.2134e-1 (3.6502e-3)	2.3579e-1 (5.6816e-2)	4.4141e-1 (1.1285e-1)	3.9811e-1 (8.8914e-2)	4.1148e-1 (3.5020e-2)	**5.5528e-1** (1.2653e-3)	5.5485e-1 (1.1195e-3)
LLM	**2.5136e-1** (2.4924e-2) +	**3.3077e-1** (5.8087e-2) +	**5.1065e-1** (9.9094e-2) +	**4.6969e-1** (1.1755e-1) +	**4.5059e-1** (6.5835e-2) +	5.4663e-1 (2.6050e-2) −	**5.5506e-1** (1.4957e-3) =
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	2.4809e-3 (2.3330e-3)	2.5142e-1 (2.5710e-3)	**2.0834e-1** (2.5761e-4)	1.5889e-1 (4.1009e-2)	**2.3216e-1** (1.1380e-1)	**6.4795e-2** (2.9266e-2)	2.8872e-1 (2.3954e-4)
LLM	**1.1932e-2** (3.0557e-2) +	**2.5657e-1** (5.5160e-3) +	2.0808e-1 (2.0499e-2) −	**1.6691e-1** (5.2022e-2) +	2.2282e-1 (1.0729e-1) =	5.6608e-2 (1.7439e-2) =	**2.8874e-1** (2.2367e-4) =
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	2.0755e-1 (2.5268e-4)	1.1593e-1 (7.6759e-3)	6.0053e-1 (3.1126e-2)	3.3566e-1 (1.1647e-1)	8.6155e-1 (6.6068e-3)	**7.8116e-1** (2.6590e-2)	4.1828e-1 (9.1092e-4)
LLM	**2.0758e-1** (3.4449e-4) =	**1.3280e-1** (1.9260e-2) +	**6.2212e-1** (1.2509e-1) =	**3.9282e-1** (2.8238e-5) +	**8.6374e-1** (6.9470e-3) =	7.4362e-1 (1.1052e-1) =	**4.3494e-1** (2.7084e-3) +
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	2.7729e-1 (7.9829e-6)	8.5393e-2 (4.7130e-3)	4.1452e-1 (2.3073e-2)	7.2755e-2 (1.0709e-2)	5.4302e-1 (2.6011e-4)	2.1071e-1 (3.9864e-2)	4.0519e-2 (1.2901e-6)
LLM	**2.7729e-1** (7.4752e-6) +	**9.1693e-2** (3.1704e-3) +	**4.6299e-1** (7.7389e-2) =	**7.7274e-2** (1.0109e-2) +	**5.4363e-1** (6.7001e-5) +	**2.3151e-1** (4.7896e-2) +	**4.0521e-2** (1.8593e-6) +

Table 12: IGD results of CMOEA-MS and CMOEA-MS-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	3.6040e-1 (2.1227e-2)	3.0844e-1 (2.1653e-2)	3.4378e-1 (1.8529e-2)	3.2599e-1 (1.5835e-2)	5.0149e-1 (3.2668e-1)	6.2891e-1 (3.7134e-1)	1.4921e-1 (2.2134e-2)
LLM	**3.1800e-1** (3.1579e-2) +	**2.5919e-1** (3.3788e-2) +	**3.2826e-1** (2.6017e-2) +	**3.0204e-1** (2.5136e-2) +	**3.3888e-1** (5.3553e-1) +	**4.0105e-1** (4.8415e-1) +	**1.0935e-1** (3.1202e-2) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	2.5272e-1 (3.1623e-2)	7.4106e-1 (1.3740e-1)	4.6642e-1 (1.2765e-1)	4.6649e-1 (1.3771e-1)	4.5982e-1 (1.3100e-1)	**9.3061e-2** (1.0633e-3)	**9.5192e-2** (8.2279e-4)
LLM	**1.3375e-1** (8.7280e-2) +	**5.9936e-1** (1.7535e-1) +	**3.5817e-1** (1.5791e-1) +	**3.7315e-1** (1.8141e-1) +	**3.7030e-1** (1.9134e-1) +	9.8329e-2 (2.3724e-2) −	9.5205e-2 (7.4677e-4) =
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	7.5871e-1 (2.0765e-2)	2.8846e-1 (2.9355e-2)	3.7152e-1 (4.5188e-2)	1.4405e-1 (1.2796e-1)	**1.9911e-1** (1.9178e-1)	**5.0870e-1** (1.1674e-1)	3.1570e-2 (6.0271e-4)
LLM	**7.1437e-1** (1.0703e-1) +	**2.4578e-1** (2.2717e-2) +	**3.4644e-1** (3.9702e-2) +	**9.1290e-2** (1.2909e-1) +	2.1115e-1 (1.7842e-1) =	5.2833e-1 (7.1579e-2) =	**3.0451e-2** (5.6035e-4) +
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	4.0242e-2 (8.8118e-4)	4.3965e-1 (4.7376e-2)	**3.6175e+5** (1.9399e+3)	**7.8702e+1** (2.1482e+1)	2.6020e+3 (1.1060e+4)	1.7183e+1 (4.2160e+0)	1.8891e+0 (1.8614e-4)
LLM	**3.9121e-2** (9.9702e-4) +	**3.6663e-1** (9.8447e-2) +	3.6228e+5 (1.9904e+3) =	9.1459e+1 (5.4893e-3) −	**4.6011e+2** (1.0565e+3) +	**1.6599e+1** (1.4160e+1) =	1.8891e+0 (1.3550e-15) =
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	6.0558e+2 (4.2749e-1)	2.5807e+6 (1.5763e+4)	2.6681e+2 (1.4941e+1)	2.0290e+3 (5.1549e+2)	4.2865e+3 (3.5585e+3)	7.0691e+3 (2.0165e+3)	**9.4508e-2** (1.0036e-4)
LLM	**6.0545e+2** (3.2616e-2) +	**2.5481e+6** (3.9696e+4) +	**1.7510e+2** (1.2339e+2) +	**1.4228e+3** (6.9667e+2) +	**2.4363e+2** (2.2044e+2) +	**5.0619e+3** (1.6951e+2) +	9.4510e-2 (9.6653e-5) =

Table 13: HV results of MOEA/D-DAE and MOEA/D-DAE-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	1.3444e-1 (2.1391e-2)	2.3457e-1 (1.8987e-2)	1.2322e-1 (1.5505e-2)	1.9119e-1 (1.4932e-2)	1.0741e-1 (1.1666e-2)	8.9014e-2 (9.9114e-3)	2.2102e-1 (5.4159e-3)
LLM	**1.8855e-1** (1.7568e-2) +	**3.4079e-1** (9.0171e-3) +	**1.6706e-1** (1.5396e-2) +	**2.5157e-1** (2.9192e-2) +	**1.7005e-1** (2.1504e-2) +	**1.3833e-1** (9.8452e-3) +	**2.4978e-1** (9.3833e-3) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	2.1743e-1 (6.4623e-3)	4.2029e-1 (8.4752e-2)	**5.6249e-1** (1.2680e-1)	**5.6238e-1** (1.3799e-1)	**5.5260e-1** (3.3799e-2)	5.5448e-1 (1.1698e-3)	**5.5242e-1** (8.4986e-4)
LLM	**2.5674e-1** (6.0759e-3) +	**4.3910e-1** (2.4444e-2) =	4.3239e-1 (6.5726e-2) −	5.3389e-1 (3.5571e-2) =	5.4912e-1 (2.6293e-2) =	**5.5527e-1** (1.2917e-3) +	5.5059e-1 (1.2059e-3) −
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	8.2406e-3 (5.0069e-3)	**2.6701e-1** (1.2182e-2)	1.5087e-1 (6.3811e-2)	**1.9872e-1** (1.2324e-2)	3.4562e-1 (1.5874e-2)	2.9356e-1 (1.5336e-2)	**2.8228e-1** (3.2417e-3)
LLM	**1.0097e-2** (4.7055e-3) =	2.6505e-1 (8.6668e-3) =	**2.1374e-1** (9.5071e-3) +	1.7113e-1 (1.5267e-2) −	**3.4759e-1** (1.1894e-3) +	**2.9943e-1** (3.4696e-2) +	2.8167e-1 (1.7496e-3) −
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	1.9042e-1 (2.1358e-3)	9.2591e-2 (3.8699e-2)	5.6823e-1 (8.0197e-3)	**8.8055e-2** (1.2676e-1)	**9.0158e-1** (8.2552e-5)	**8.5912e-1** (5.9589e-4)	4.3496e-1 (1.1114e-4)
LLM	**1.9944e-1** (2.5498e-3) +	**1.3147e-1** (4.5629e-2) +	**5.8700e-1** (7.1555e-3) +	6.5270e-2 (1.2083e-1) =	9.0107e-1 (2.4959e-3) =	8.5810e-1 (1.9255e-3) −	**4.3507e-1** (1.1241e-4) +
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	2.7690e-1 (1.2648e-4)	9.7558e-2 (5.5768e-4)	5.5176e-1 (4.7314e-3)	8.8335e-2 (6.0095e-5)	4.4552e-1 (2.4135e-2)	1.6670e-1 (4.5278e-2)	4.0504e-2 (4.3292e-6)
LLM	**2.7693e-1** (1.3864e-4) =	**9.8269e-2** (4.0641e-4) +	**5.5309e-1** (3.0690e-3) =	**8.8357e-2** (7.3981e-5) =	**4.9631e-1** (1.7677e-2) +	**1.8975e-1** (4.5040e-2) +	**4.0507e-2** (4.2544e-6) +

Table 14: IGD results of MOEA/D-DAE and MOEA/D-DAE-LLM on the 35 test problems.

Algorithm	LIR-CMOP1	LIR-CMOP2	LIR-CMOP3	LIR-CMOP4	LIR-CMOP5	LIR-CMOP6	LIR-CMOP7
Ori.	4.0860e-1 (1.4444e-1)	2.0654e-1 (3.7980e-2)	3.3851e-1 (9.9771e-2)	2.6585e-1 (2.5998e-2)	4.2830e-1 (3.4870e-2)	5.6673e-1 (7.7147e-2)	2.0800e-1 (2.3008e-2)
LLM	**1.1762e-1** (5.8916e-2) +	**3.9565e-2** (1.5070e-2) +	**1.1461e-1** (5.9152e-2) +	**1.4892e-1** (6.3905e-2) +	**2.3962e-1** (5.8783e-2) +	**2.7017e-1** (5.4811e-2) +	**1.1675e-1** (3.1622e-2) +
Algorithm	LIR-CMOP8	LIR-CMOP9	LIR-CMOP10	LIR-CMOP11	LIR-CMOP12	LIR-CMOP13	LIR-CMOP14
Ori.	2.3435e-1 (2.2504e-2)	4.1663e-1 (2.1858e-1)	**2.1784e-1** (1.7400e-1)	3.5416e-1 (3.7826e-1)	2.0513e-1 (1.4706e-1)	9.8105e-2 (7.9627e-4)	**1.0027e-1** (7.8402e-4)
LLM	**9.5084e-2** (2.0604e-2) +	**3.6235e-1** (4.8269e-2) =	3.4197e-1 (8.7484e-2) −	**2.5956e-1** (6.0208e-2) =	**1.4135e-1** (7.6787e-2) =	**9.7231e-2** (8.5352e-4) +	1.0065e-1 (8.6672e-4) =
Algorithm	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7
Ori.	7.2209e-1 (2.7024e-2)	**2.0906e-1** (3.3599e-2)	4.5021e-1 (1.5276e-1)	**2.7112e-2** (6.9486e-2)	1.0689e-2 (2.4425e-2)	5.8629e-2 (4.1375e-2)	4.5754e-2 (9.0159e-3)
LLM	**7.1264e-1** (2.4396e-2) =	2.2007e-1 (2.5787e-2) =	**3.0259e-1** (5.2093e-2) +	1.8861e-1 (1.0194e-1) −	**6.8879e-3** (1.4959e-3) +	**5.3006e-2** (9.5652e-2) +	**4.3176e-2** (3.9272e-3) =
Algorithm	DAS-CMOP8	DAS-CMOP9	RWMOP1	RWMOP2	RWMOP3	RWMOP4	RWMOP5
Ori.	8.7296e-2 (6.4586e-3)	5.4910e-1 (1.7157e-1)	**3.6060e+5** (1.0068e+3)	1.6327e+2 (1.7426e+2)	**2.6065e+3** (2.1169e+3)	**4.4558e-1** (3.4078e-1)	1.8883e+0 (2.0908e-4)
LLM	**6.5368e-2** (9.5899e-3) +	**3.6272e-1** (2.2549e-1) +	3.6210e+5 (8.6721e+3) =	**1.1401e+2** (9.2345e+1) =	3.4350e+3 (8.0414e+3) =	5.1665e-1 (6.3924e-1) =	**1.8881e+0** (4.6490e-4) +
Algorithm	RWMOP6	RWMOP11	RWMOP12	RWMOP13	RWMOP15	RWMOP17	RWMOP18
Ori.	2.3324e+3 (2.3713e+2)	**2.5233e+6** (3.0124e+4)	1.9707e+0 (8.1367e-1)	**3.7037e+2** (9.7319e+0)	9.1032e+3 (5.8642e+3)	1.0324e+4 (6.9038e+3)	9.4550e-2 (3.6673e-5)
LLM	**2.1908e+3** (4.1633e+2) =	2.5328e+6 (3.3000e+4) =	**1.9463e+0** (2.4060e+0) =	3.9163e+2 (4.5205e+1) −	**4.2262e+3** (3.3919e+3) +	**4.9945e+3** (2.6802e+3) +	**9.4507e-2** (1.1584e-4) =

F Convergence and Solution Distributions

F.1 Convergence Curves

Figures 9, 10, 11, 12, 13, 14, and 15 compare the HV and IGD convergence trajectories of the original CMOEAs and their LLM-enhanced variants. Overall, the enhanced variants show better convergence on most problems. On LIR-CMOP, several original algorithms stagnate during the middle or later search stages, whereas the corresponding enhanced variants continue to improve; this behavior is evident on the representative LIR-CMOP5 and LIR-CMOP10 cases and on the additional LIR-CMOP1–4 and LIR-CMOP7–12 cases. The differences become smaller on LIR-CMOP13 and LIR-CMOP14, where several original algorithms already achieve competitive indicator values. The results on DAS-CMOP are more problem dependent, with clear improvements on DAS-CMOP1, DAS-CMOP3, and DAS-CMOP9 but limited improvement on DAS-CMOP5. Several RWMOP cases also show faster or better convergence, although many trajectories remain close to those of the original algorithms. These results indicate that LLM-guided reconstruction is most beneficial when the current variation behavior loses search efficiency, while retaining the original operator preserves stability when the original search remains effective.

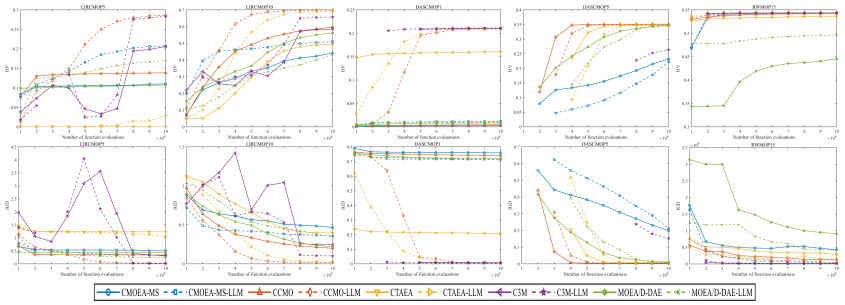


Figure 9: Convergence on representative problems

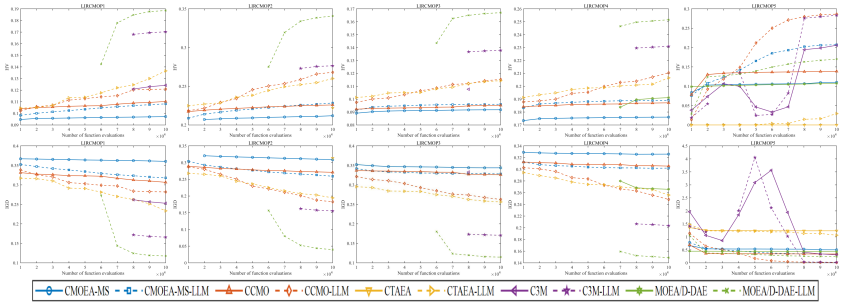


Figure 10: Convergence on LIR-CMOP1–5

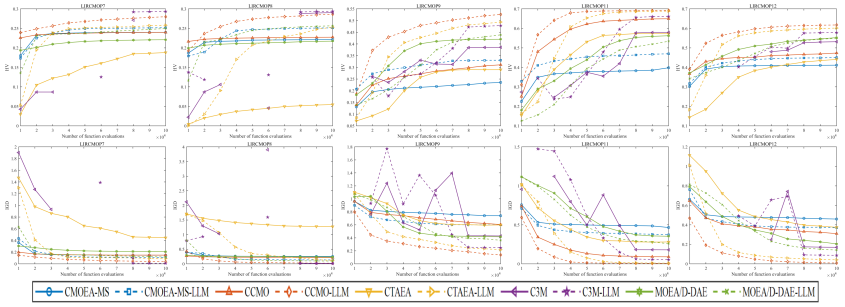


Figure 11: Convergence on LIR-CMOP7–9 and LIR-CMOP11–12

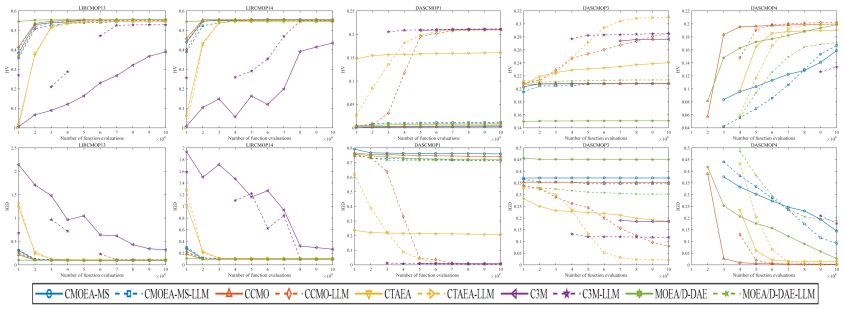


Figure 12: Convergence on LIR-CMOP13–14 and DAS-CMOP1, 3, and 4

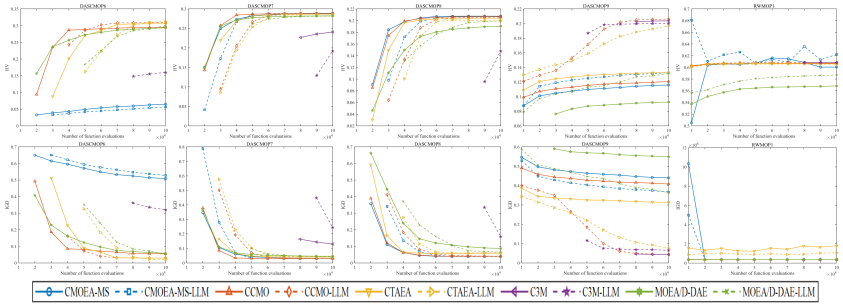


Figure 13: Convergence on DAS-CMOP6–9 and RWMOP1

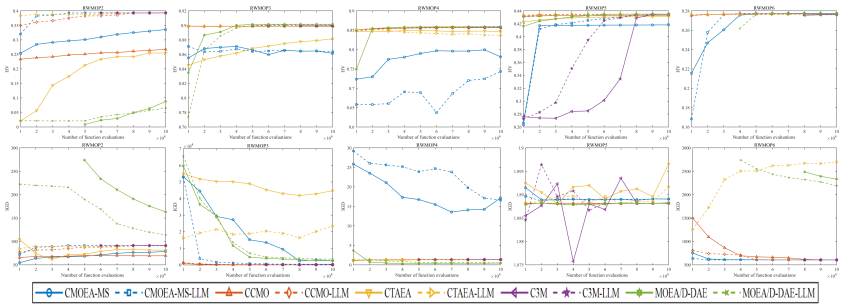


Figure 14: Convergence on RWMOP2–6

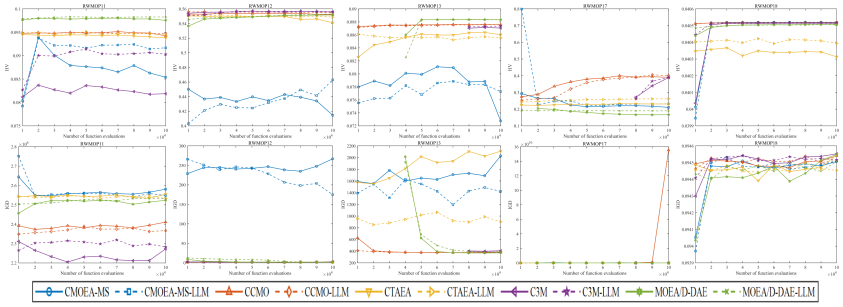


Figure 15: Convergence on RWMOP11–13 and RWMOP17–18

F.2 SOLUTION DISTRIBUTIONS

Figures 16, 17, 18, 19, 20, 21, and 22 compare the final solution distributions obtained by CCMO and CCMO-LLM. Overall, CCMO-LLM provides broader and more continuous front coverage on several problems. This improvement is evident on the representative LIR-CMOP5, LIR-CMOP6, LIR-CMOP10, DAS-CMOP1, and DAS-CMOP2 cases and in the additional results for LIR-CMOP1–5, LIR-CMOP7–9, and LIR-CMOP12, where the original CCMO often covers only a restricted part of the feasible front. In contrast, both methods produce well-distributed approximations on LIR-CMOP13 and LIR-CMOP14. The effect on DAS-CMOP varies across problems, with a clear extension of front coverage on DAS-CMOP1 and smaller changes on several other cases. The RWMOP distributions are generally more similar, although differences in coverage and concentration remain visible on some problems. These observations are consistent with the statistical results and indicate that LLM-guided operator reconstruction mainly improves exploration when the original operator misses parts of the feasible front, while largely preserving an already well-distributed approximation.

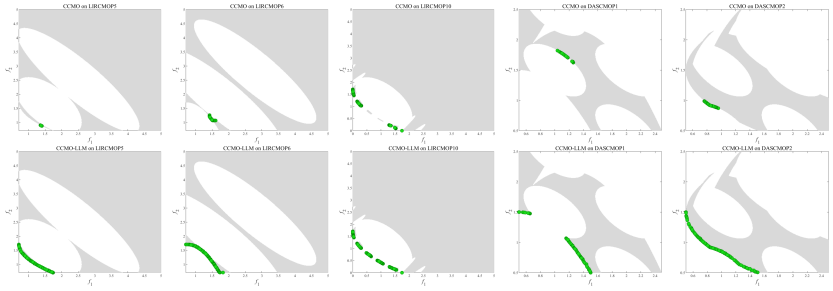


Figure 16: Distributions on representative problems

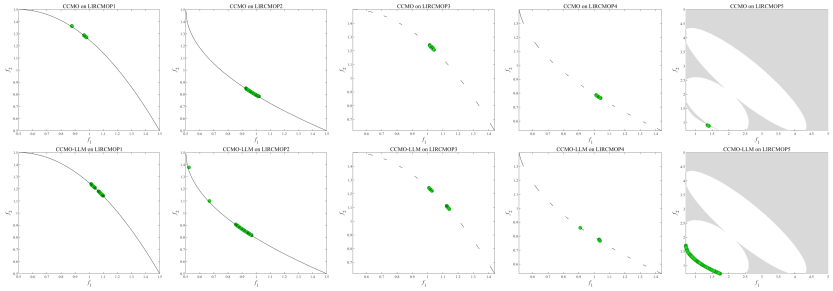


Figure 17: Distributions on LIR-CMOP1–5

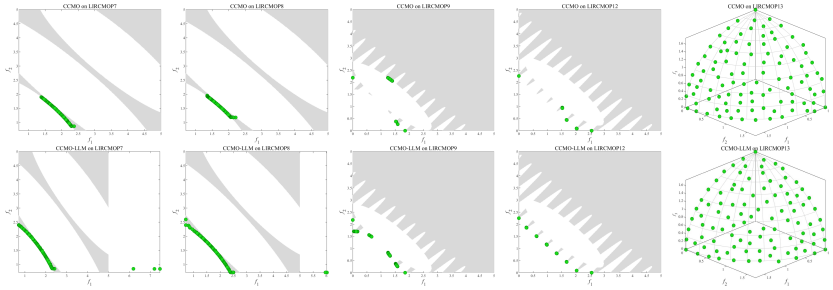


Figure 18: Distributions on LIR-CMOP7–9 and LIR-CMOP12–13

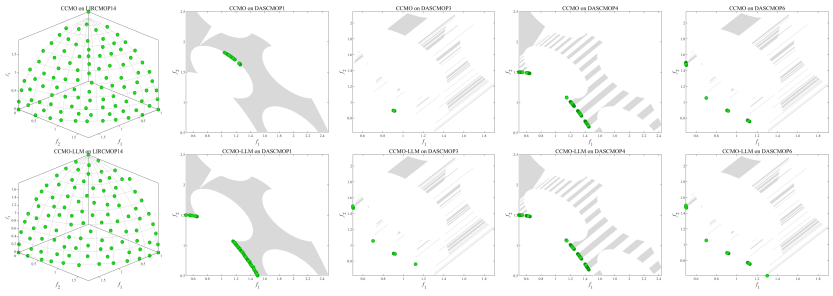


Figure 19: Distributions on LIR-CMOP14 and DAS-CMOP1, 3, 4, and 6

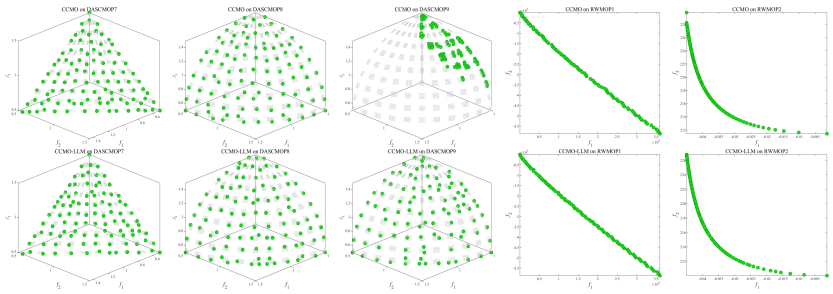


Figure 20: Distributions on DAS-CMOP7–9 and RWMOP1–2

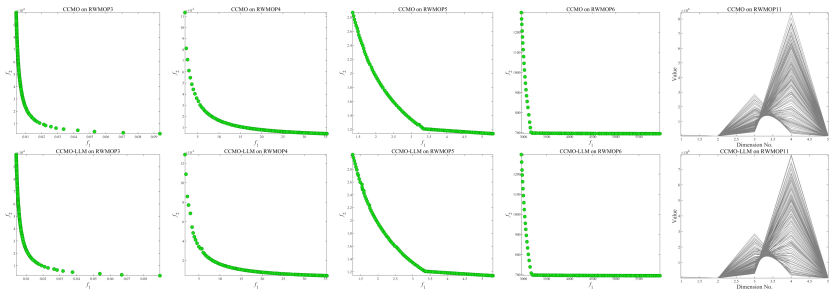


Figure 21: Distributions on RWMOP3–6 and RWMOP11

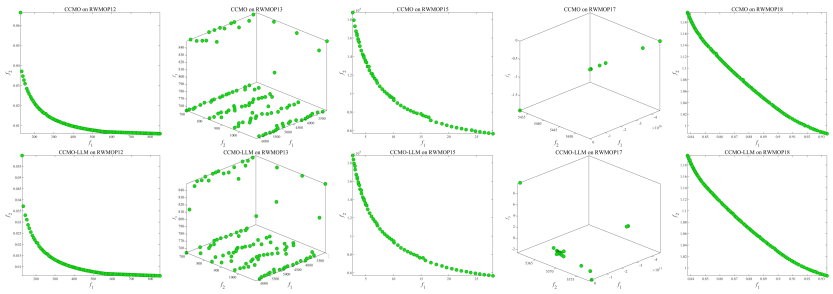


Figure 22: Distributions on RWMOP12–13, RWMOP15, and RWMOP17–18

G Parameter Sensitivity Analysis

Eight parameter configurations are considered for CMOEA-MS-LLM. The default CMOEA-MS-LLM uses $\delta = 0.12$ and $\gamma = 0.75$. To evaluate the effect of δ, γ is fixed at 0.75 and δ is varied among 0.06, 0.09, 0.12, 0.15, and 0.18. To evaluate the effect of γ, δ is fixed at 0.12 and γ is varied among 0.25, 0.50, 0.75, and 1.00. Table 15 lists the default configuration and the seven parameter variants. Tables 16 and 17 report the problem-level Friedman ranks for IGD and HV, respectively. Lower ranks indicate better performance.

Table 15: Parameter settings of CMOEA-MS-LLM and its variants.

Variant	δ	γ
CMOEA-MS-LLM	0.12	0.75
CMOEA-MS-LLM$_{\delta=0.06}$	0.06	0.75
CMOEA-MS-LLM$_{\delta=0.09}$	0.09	0.75
CMOEA-MS-LLM$_{\delta=0.15}$	0.15	0.75
CMOEA-MS-LLM$_{\delta=0.18}$	0.18	0.75
CMOEA-MS-LLM$_{\gamma=0.25}$	0.12	0.25
CMOEA-MS-LLM$_{\gamma=0.50}$	0.12	0.50
CMOEA-MS-LLM$_{\gamma=1.00}$	0.12	1.00

The problem-level results further support the overall trend observed in Table 3. The default CMOEA-MS-LLM obtains the lowest IGD rank on six of the nine problems and the lowest or tied-lowest HV rank on seven problems. However, it is not optimal for every individual problem. CMOEA-MS-LLM$_{\delta=0.18}$ performs best on DAS-CMOP1, while CMOEA-MS-LLM$_{\delta=0.06}$ performs best on DAS-CMOP9. Among the alternative settings of δ, CMOEA-MS-LLM$_{\delta=0.06}$ gives the closest overall performance to the default configuration. For γ, CMOEA-MS-LLM$_{\gamma=0.25}$ produces the highest average ranks for both indicators. Increasing γ to 0.50 improves the overall performance, while the default value of 0.75 achieves the lowest average ranks. Further increasing γ to 1.00 provides no additional benefit. Overall, the selected $(\delta = 0.12, \gamma = 0.75)$ setting provides robust performance across the tested problems rather than being optimal for every individual problem.

Table 16: Problem-level Friedman ranks of CMOEA-MS-LLM and its parameter variants for IGD.

Variant	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7	DAS-CMOP8	DAS-CMOP9	Average
CMOEA-MS-LLM	4.533	**3.567**	**3.467**	**3.400**	**3.433**	3.900	**3.700**	**3.433**	3.767	**3.6889**
CMOEA-MS-LLM$_{\delta=0.06}$	4.233	4.667	5.133	3.967	4.133	4.833	4.433	4.233	**3.733**	4.3741
CMOEA-MS-LLM$_{\delta=0.09}$	4.567	4.600	4.300	4.300	5.267	4.967	5.067	4.800	4.367	4.6926
CMOEA-MS-LLM$_{\delta=0.15}$	4.633	4.400	4.533	5.083	4.300	4.583	4.267	5.033	4.133	4.5519
CMOEA-MS-LLM$_{\delta=0.18}$	**3.633**	4.133	5.200	5.583	4.733	4.817	4.300	4.867	5.300	4.7296
CMOEA-MS-LLM$_{\gamma=0.25}$	4.933	5.400	4.000	4.867	5.500	**3.867**	4.800	4.533	4.633	4.7259
CMOEA-MS-LLM$_{\gamma=0.50}$	4.967	4.500	4.533	3.933	4.400	4.233	4.967	4.600	5.033	4.5741
CMOEA-MS-LLM$_{\gamma=1.00}$	4.500	4.733	4.833	4.867	4.233	4.800	4.467	4.500	5.033	4.6630

Table 17: Problem-level Friedman ranks of CMOEA-MS-LLM and its parameter variants for HV.

Variant	DAS-CMOP1	DAS-CMOP2	DAS-CMOP3	DAS-CMOP4	DAS-CMOP5	DAS-CMOP6	DAS-CMOP7	DAS-CMOP8	DAS-CMOP9	Average
CMOEA-MS-LLM	4.350	**4.100**	**3.800**	**3.533**	**3.467**	**3.933**	**3.633**	**4.100**	4.033	**3.8833**
CMOEA-MS-LLM$_{\delta=0.06}$	4.183	4.500	4.833	3.933	4.067	4.800	5.267	4.367	**3.533**	4.3870
CMOEA-MS-LLM$_{\delta=0.09}$	4.633	4.800	3.967	4.333	5.267	4.833	4.400	4.433	4.567	4.5815
CMOEA-MS-LLM$_{\delta=0.15}$	4.567	4.133	4.900	5.117	4.400	4.583	4.633	4.333	4.567	4.5815
CMOEA-MS-LLM$_{\delta=0.18}$	**3.717**	4.133	5.267	5.483	4.767	4.817	4.067	5.167	5.000	4.7130
CMOEA-MS-LLM$_{\gamma=0.25}$	5.050	5.367	4.200	4.733	5.367	**3.933**	5.367	4.167	4.400	4.7315
CMOEA-MS-LLM$_{\gamma=0.50}$	4.983	4.533	4.600	4.000	4.333	4.233	4.500	4.800	4.967	4.5500
CMOEA-MS-LLM$_{\gamma=1.00}$	4.517	4.433	4.433	4.867	4.333	4.867	4.133	4.633	4.933	4.5722

H MOOPF FORMULATION AND TEST CASES

H.1 CONTROL VARIABLES AND TEST CASES

The MOOPF test cases are adapted from Zhu et al. (2025). The decision vector comprises active power outputs of nonslack generators, generator voltage magnitudes, transformer tap ratios, and shunt compensation settings. The IEEE 30-bus system contains 5, 6, 4, and 9 variables of these types, respectively, whereas the IEEE 57-bus system contains 6, 7, 17, and 3. The slack generator active power, generator reactive power outputs, load bus voltages, and branch power flows are determined by AC power flow calculations rather than optimized independently.

Generator voltage magnitudes are restricted to 0.95–1.10 p.u. Transformer tap ratios range from 0.90 to 1.10 with a spacing of 0.02. Shunt compensation settings are discretized in increments of 1 MVAr, with ranges of 0–5 MVAr and 0–20 MVAr for the IEEE 30-bus and 57-bus systems, respectively. Although the decision vectors use a real representation, tap ratios and compensation settings are mapped to their permitted discrete values before power flow evaluation.

H.2 OBJECTIVE FUNCTIONS

Four objectives characterize the economic, environmental, and voltage regulation requirements of the system. Generation fuel cost is represented by a quadratic function of generator active power output

$$f_{\text{cost}} = \sum_{g \in \mathcal{G}_{\text{gen}}} \left(a_g + b_g P_g + c_g P_g^2\right) \tag{21}$$

where $\mathcal{G}_{\text{gen}}$ is the generator set, P_g is the active power output of generator g in MW, and a_g, b_g, and c_g are its fuel cost coefficients.

Active power loss is calculated from the power injections at both ends of each branch

$$f_{\text{loss}} = \sum_{\ell \in \mathcal{L}_{\text{br}}} \left(P_\ell^{\text{from}} + P_\ell^{\text{to}}\right) \tag{22}$$

where $\mathcal{L}_{\text{br}}$ is the branch set, and P_ℓ^{from} and P_ℓ^{to} are the active power injections into branch ℓ from its two terminal buses.

Pollutant emissions are evaluated using the coefficient convention and power scaling adopted in the implementation

$$f_{\text{emission}} = \sum_{g \in \mathcal{G}_{\text{gen}}} \left[10^{-2}\left(\alpha_g + \beta_g p_g + \gamma_g p_g^2\right) + \xi_g \exp\left(\lambda_g p_g\right)\right] \tag{23}$$

where $p_g = P_g/(100\,\text{MW})$, and α_g, β_g, γ_g, ξ_g, and λ_g are the emission coefficients of generator g.

Load voltage deviation measures the aggregate departure of load bus voltage magnitudes from 1 p.u.

$$f_{\text{voltage}} = \sum_{i \in \mathcal{N}_L} |V_i - 1| \tag{24}$$

where $\mathcal{N}_L$ is the set of nongenerator buses used for voltage deviation evaluation, and V_i is the voltage magnitude at bus i in per unit. Table 18 lists the objective combinations minimized in the eight cases.

Table 18: MOOPF test cases

Case	System	Objectives
MOOPF1	IEEE 30	f_{cost}, f_{loss}
MOOPF2	IEEE 30	f_{cost}, f_{emission}
MOOPF3	IEEE 30	f_{cost}, f_{loss}, f_{emission}
MOOPF4	IEEE 30	f_{cost}, f_{loss}, f_{emission}, f_{voltage}
MOOPF5	IEEE 57	f_{cost}, f_{loss}
MOOPF6	IEEE 57	f_{cost}, f_{emission}
MOOPF7	IEEE 57	f_{cost}, f_{loss}, f_{emission}
MOOPF8	IEEE 57	f_{cost}, f_{loss}, f_{emission}, f_{voltage}

H.3 Operating Constraints and Evaluation

Candidate control settings are incorporated into the network model before solving the AC power flow equations. A converged solution satisfies the power balance equations to the solver tolerance. Feasibility additionally requires generator active and reactive power outputs, generator and load bus voltages, and control settings to remain within their prescribed limits. Load bus voltage limits are 0.95–1.05 p.u. for the IEEE 30-bus system and 0.94–1.06 p.u. for the IEEE 57-bus system.

Branch loading is evaluated at both terminals, and the larger apparent power magnitude is compared with the branch rating

$$\max\left\{\left|S_\ell^{\mathrm{from}}\right|, \left|S_\ell^{\mathrm{to}}\right|\right\} \le S_\ell^{\max} \tag{25}$$

where S_ℓ^{from} and S_ℓ^{to} are the complex power injections at the two terminals, and $S_\ell^{\max}$ is the apparent power limit.

The IEEE 30-bus implementation imposes capacity limits on all 41 branches. For the IEEE 57-bus system, branch capacity constraints are included only where positive ratings are supplied by the case data. Generator reactive power limits are checked after the power flow calculation without enabling automatic reactive limit enforcement. Each operating constraint is returned separately in the form $g_j \le 0$, allowing the original CMOEA to apply its own constraint handling mechanism. If the power flow solver fails to converge, all objective and constraint entries are assigned 10^6, marking the candidate as infeasible.

Before returning objective values to the optimizer, each selected objective is scaled using fixed case-specific constants

$$\widetilde{f}_j = \frac{f_j - L_j}{U_j - L_j} \tag{26}$$

where L_j and U_j are fixed scaling constants rather than estimated Pareto front bounds. Values outside the scaling interval are not clipped.

H.4 Ablation Results on MOOPF Problems

Table 19 shows that the complete CMOEA-MS-LLM achieves the best mean IGD^+ and HV values on MOOPF2, MOOPF3, MOOPF5, MOOPF6, and MOOPF8. The ablation variants remain competitive on several individual cases. CMOEA-MS-LLM-AW obtains the best mean results on MOOPF1 and MOOPF4, while CMOEA-MS-LLM-LR achieves the lowest IGD^+ on MOOPF7. However, these advantages are not consistent across both metrics or across different problems. In particular, removing the guided search directions leads to noticeable degradation on several cases. Overall, the problem-level results are consistent with the Friedman ranks in Figure 4, where the complete framework achieves the best overall performance.

Table 19: Ablation results on the MOOPF problems.

Problem	Metric	CMOEA-MS-LLM-GD	CMOEA-MS-LLM-AW	CMOEA-MS-LLM-LR	CMOEA-MS-LLM
MOOPF1	IGD^+	1.1378e-1 (8.32e-2) −	**4.3962e-2** (8.75e-2) =	5.0796e-2 (9.46e-2) =	7.3798e-2 (1.05e-1)
	HV	5.8061e-1 (1.43e-1) −	**7.2316e-1** (1.55e-1) =	7.1055e-1 (1.65e-1) =	6.7067e-1 (1.86e-1)
MOOPF2	IGD^+	1.5148e-3 (7.09e-5) =	1.5455e-3 (1.79e-4) =	1.5503e-3 (1.23e-4) −	**1.4997e-3** (9.98e-5)
	HV	8.3377e-1 (5.89e-4) =	8.3351e-1 (7.82e-4) −	8.3362e-1 (5.33e-4) −	**8.3393e-1** (8.45e-4)
MOOPF3	IGD^+	2.2304e-2 (1.33e-2) −	2.2820e-2 (1.82e-2) −	2.0890e-2 (1.34e-2) −	**1.2165e-2** (8.71e-3)
	HV	6.4557e-1 (4.76e-2) −	6.4136e-1 (6.23e-2) −	6.5053e-1 (4.82e-2) −	**6.7965e-1** (3.43e-2)
MOOPF4	IGD^+	2.5326e-2 (1.35e-3) =	**2.4439e-2** (9.24e-4) +	2.5703e-2 (2.10e-3) =	2.5478e-2 (1.83e-3)
	HV	6.4704e-1 (3.13e-3) =	**6.4812e-1** (2.82e-3) =	6.4644e-1 (4.53e-3) =	6.4686e-1 (3.22e-3)
MOOPF5	IGD^+	1.8934e-2 (1.54e-2) =	1.9703e-2 (1.46e-2) =	1.8570e-2 (8.80e-3) =	**1.5385e-2** (8.11e-3)
	HV	7.9907e-1 (7.22e-2) =	7.7973e-1 (7.08e-2) −	8.0292e-1 (5.13e-2) =	**8.1769e-1** (3.05e-2)
MOOPF6	IGD^+	3.9813e-3 (7.91e-4) −	4.3374e-3 (6.39e-4) −	3.9346e-3 (8.77e-4) =	**3.5806e-3** (1.29e-3)
	HV	8.0336e-1 (2.24e-3) =	8.0209e-1 (1.71e-3) −	8.0355e-1 (2.57e-3) =	**8.0446e-1** (4.01e-3)
MOOPF7	IGD^+	1.4659e-1 (5.55e-2) −	1.2442e-1 (6.64e-2) =	**1.1693e-1** (4.02e-2) =	1.2555e-1 (8.14e-2)
	HV	3.0808e-1 (6.14e-2) −	3.4435e-1 (6.44e-2) =	3.5124e-1 (6.13e-2) =	**3.6699e-1** (1.12e-1)
MOOPF8	IGD^+	5.6539e-2 (2.30e-2) =	4.9329e-2 (1.83e-2) =	5.7525e-2 (2.50e-2) −	**4.7218e-2** (1.96e-2)
	HV	4.8069e-1 (4.90e-2) −	4.8299e-1 (3.03e-2) −	4.6953e-1 (4.10e-2) −	**5.0846e-1** (4.71e-2)
+/ = /−	IGD^+	0/4/4	1/5/2	0/5/3	—
+/ = /−	HV	0/4/4	0/3/5	0/5/3	—