HYPAEVO: CAPABILITY-ALIGNED LLM SCORING FOR METAHEURISTIC HYPERPARAMETER CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic hyperparameter control is a long-standing challenge for population-based metaheuristics. Learning-based controllers achieve strong performance but require expensive training and generalize poorly. Recent zero-shot approaches use Large Language Models (LLMs) to bypass training, but existing architectures route the final control decision through open-ended numerical generation (a regime in which LLMs are comparatively less reliable) while leaving their stronger capabilities in ordinal assessment underexploited. We propose **HypaEvo**, a *capability-aligned dual-LLM scoring architecture* that confines the LLM to bounded ordinal judgment and delegates numerical actuation to deterministic code. Two specialized agents independently rate the urgency of exploration and exploitation on a discrete ordinal scale (0–10), whereas a deterministic coordinator translates the score differential into proportional, range-normalized parameter adjustments via a fixed direction map, enforcing directional priors and parameter bounds by construction. This separates qualitative reasoning from numerical arithmetic, removing open-ended numerical generation from the actuation step. Across four combinatorial optimization problems (TSP, CVRP, FSSP, and UAV trajectory optimization) and three metaheuristic families (GA, PSO, and ACO) with Qwen3-30B as the backbone, our method consistently outperforms representative state-of-the-art learning-based, neural-evolution, and LLM-driven controllers.

1 INTRODUCTION

Population-based metaheuristics such as genetic algorithms, particle swarm optimization, and ant colony optimization are widely used for combinatorial and black-box optimization (Eiben et al., 1999; Dorigo et al., 2006; Kennedy & Eberhart, 1995). Their performance depends on the balance between exploration and exploitation, mediated by hyperparameters whose preferred values can change as search progresses (Karafotias et al., 2015). Dynamically controlling these hyperparameters is therefore important, yet remains difficult to automate reliably.

Large Language Models (LLMs) have recently emerged as a promising zero-shot mechanism for algorithm control (Xu et al., 2026), using semantic priors from scientific text and code (Liu et al., 2024a; Ye et al., 2024) to reason about convergence, diversity, and stagnation. We focus on a mismatch between this qualitative reasoning capability and the actuation interface of current LLM-driven controllers. LLM-as-judge studies show that LLMs can be effective *ordinal assessors* for rating, ranking, and comparison (Zheng et al., 2023; Liu et al., 2023b), yet existing controllers still route the final actuation step through numerical generation (Xu et al., 2026). After identifying a desirable search direction, the model must still commit to absolute parameter values, adding a numerical calibration burden. Moreover, heterogeneous diversity, fitness-quality, and temporal signals can be resolved inside a single controller rather than exposed as separable signals (Xu et al., 2026), while known directional priors among hyperparameters remain prompt-level instructions with no architectural enforcement.

Motivated by this asymmetry, we formalize a *dual-LLM scoring architecture* that decomposes hyperparameter control into two stages: ordinal assessment of search-state urgency by specialized LLM agents, followed by deterministic numerical actuation through a closed-form update rule. This delegates qualitative assessment to LLMs and arithmetic, constraint enforcement, and proportional mapping to deterministic code. It further provides structural guarantees, such as enforcement of

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.

directional priors and parameter bounds, that prior architectures leave to LLM compliance. Our backbone study further shows that the scoring interface remains effective across the tested LLM backbones.

We instantiate this principle as **HypaEvo**, a dual-LLM framework for population-based metaheuristics. An *Explorer* scores exploration urgency from diversity features and an *Exploiter* scores exploitation urgency from fitness-quality features, each on a 0–10 ordinal scale. A deterministic *coordinator* converts the score differential into proportional, range-normalized hyperparameter updates through a fixed direction map with adaptive step sizing and late-stage damping. The system requires exactly two LLM calls per decision point regardless of the number of controlled hyperparameters and exposes the dual scores as interpretable intermediate signals.

Across four representative problem classes (TSP, CVRP, FSSP, and UAV trajectory optimization) and three metaheuristic families (GA, PSO, and ACO), HypaEvo improves over matched controller baselines in solution quality. On the original TSP study, it also runs substantially faster than AutoEP (Xu et al., 2026), a state-of-the-art LLM-driven controller and the closest prior method to our setting. Across the tested backbones, the scoring interface remains effective.

Contributions. Our principal contributions are fivefold. ① We identify a *capability mismatch* in current LLM-driven controllers: final decisions rely on open-ended numerical generation while bounded ordinal assessment remains underused (Zheng et al., 2023; Liu et al., 2023b). ② We propose a *capability-aligned design principle* that assigns qualitative assessment to LLMs and numerical actuation to deterministic code, enabling structural guarantees by construction. ③ We instantiate this principle as **HypaEvo**, with specialized exploration and exploitation assessors and a deterministic coordinator that maps their score differential to directional hyperparameter updates. ④ We introduce a *structured feature taxonomy* that separates diversity, fitness-quality, and temporal signals while correcting representation-specific measurement issues in combinatorial problems. ⑤ Across four problem classes and three metaheuristic families, HypaEvo improves over matched controller baselines; ablations isolate the key design components, and backbone experiments show consistent effectiveness across the tested models.

2 Related Work

Hyperparameter control for metaheuristics. Classical parameter control spans static configuration and online adaptive mechanisms (Eiben et al., 1999; Karafotias et al., 2015). Static methods such as random search, Bayesian optimization, and racing select parameters before execution (Bergstra & Bengio, 2012; Snoek et al., 2012; López-Ibáñez et al., 2016), while rule-based strategies adapt them online from search behavior (Thierens, 2005; Joshi & Bansal, 2020). More recently, deep reinforcement learning has been used to learn state-to-action control policies (Ma et al., 2024; Liu et al., 2023a; Guo et al., 2025), but requires a separate training phase and interaction budget. LLM-driven control instead offers online adaptation without controller-specific training.

LLM-driven control of metaheuristics. Offline LLM approaches such as EoH, ReEvo, and FunSearch generate algorithmic artifacts before deployment (Liu et al., 2024a; Ye et al., 2024; Romera-Paredes et al., 2024); we include them as broader context rather than matched predecessors of online hyperparameter control. Online approaches either generate candidate solutions directly (Lange et al., 2024; Yang et al., 2024) or adapt algorithm parameters. The closest prior work is AutoEP (Xu et al., 2026), which grounds compositional LLM reasoning in real-time ELA features (Mersmann et al., 2011). We retain its zero-shot, quantitatively grounded, compositional design, but change the interface between reasoning and actuation: AutoEP's final actuator emits absolute parameter values, whereas HypaEvo delegates numerical actuation to deterministic code.

LLMs as ordinal assessors. LLM-as-judge studies show that bounded rating and comparison provide effective interfaces for LLM evaluation (Zheng et al., 2023; Liu et al., 2023b). Recent work further studies pointwise versus pairwise judgments and score resolution (Godfrey et al., 2025; Tripathi et al., 2025). These findings motivate a bounded pointwise ordinal interface rather than a claim that any particular scale is universally optimal; HypaEvo uses an 11-level scale and evaluates its resolution empirically. The LLM is queried only for ordinal urgency assessment, while deterministic code handles arithmetic, bounds, and directional priors. A broader review appears in Appendix B.

3 PROBLEM FORMULATION

We formulate dynamic hyperparameter control as an online sequential decision problem under partial observation and then distinguish two interfaces for LLM-driven control.

3.1 POPULATION-BASED METAHEURISTICS AND ONLINE CONTROL

Let $\mathfrak{I}$ denote a family of problem instances. For an instance $\mathcal{I} \in \mathfrak{I}$, let $\mathcal{X}_\mathcal{I}$ denote its feasible solution space and consider $\min_{x \in \mathcal{X}_\mathcal{I}} f_\mathcal{I}(x)$, where the objective is accessible only through point evaluations. A population-based metaheuristic maintains a latent search state $s^{(t)}$, containing its population and any algorithm-specific memory, and evolves as $s^{(t+1)} = \Phi_\mathcal{M}(s^{(t)}, \boldsymbol{\lambda}^{(t+1)}; \xi^{(t)})$, where $\mathcal{M}$ is the underlying metaheuristic, $\Phi_\mathcal{M}$ its stochastic transition operator, $\boldsymbol{\lambda}^{(t+1)} \in \Lambda$ the configuration selected after observing $s^{(t)}$, and $\xi^{(t)}$ captures stochasticity. The population $P^{(t)} \subseteq \mathcal{X}_\mathcal{I}$ is a component of $s^{(t)}$ and provides population-level statistics to the observation map below; $\Lambda = \Lambda_1 \times \cdots \times \Lambda_d$ may contain continuous, integer-valued, or categorical components.

Because search dynamics are non-stationary (Eiben et al., 1999; Karafotias et al., 2015), a fixed configuration can be restrictive. Dynamic control instead varies $\boldsymbol{\lambda}^{(t)}$ online to minimize the expected terminal best-so-far objective, avoiding offline optimization of the full T-step configuration trajectory.

An observation map $\Psi : \mathcal{S} \to \mathcal{O} \subseteq \mathbb{R}^k$ compresses $s^{(t)}$ into $\mathbf{o}^{(t)} = \Psi(s^{(t)})$, describing diversity, fitness structure, and temporal progress (Mersmann et al., 2011; Ma et al., 2024). Since Ψ is many-to-one, control is partially observed. We denote by $\mathcal{H}^{(t)}$ the history of past observations, control decisions, and outcomes; HypaEvo uses the bounded sliding-window realization in Section 4.5. Let $x^{(t)}_{\text{best}}$ be the best solution observed up to time t, and define the per-step improvement $r^{(t)} = f_\mathcal{I}(x^{(t)}_{\text{best}}) - f_\mathcal{I}(x^{(t+1)}_{\text{best}}) \geq 0$. This is an outcome of the transition rather than an input to the current decision and, once observed, can enter $\mathcal{H}^{(t+1)}$. A policy uses $(\mathbf{o}^{(t)}, \mathcal{H}^{(t)})$ to adapt the configuration online. Since the reward sum telescopes to $f_\mathcal{I}(x^{(0)}_{\text{best}}) - f_\mathcal{I}(x^{(T)}_{\text{best}})$, maximizing cumulative improvement is equivalent to minimizing the expected terminal best-so-far value.

State-dependent effects of hyperparameters. Hyperparameter effects depend on the current state: increased mutation, for example, may restore diversity under stagnation but disrupt refinement near a promising basin. We therefore distinguish *realized effects*, observable only after applying a configuration, from *directional priors*, which encode dominant first-order tendencies under standard operators. HypaEvo uses these tendencies only to constrain actuation, whereas the timing and magnitude of updates are determined online.

3.2 LLM-DRIVEN CONFIGURATION POLICIES

Let $\mathcal{L}$ denote a frozen pre-trained LLM. We write the decoded LLM output as a generic control signal $\mathbf{c}^{(t)}$ and separate it from numerical actuation:

$$\begin{aligned} \mathbf{c}^{(t)} &= \mathcal{D}\Big(\mathcal{L}\big(\textsc{Prompt}(\mathbf{o}^{(t)}, \mathcal{H}^{(t)}, K)\big)\Big), \\ \boldsymbol{\lambda}^{(t+1)} &= \mathcal{A}\big(\mathbf{c}^{(t)}, \boldsymbol{\lambda}^{(t)}, t/T\big), \end{aligned} \tag{1}$$

where K contains optional algorithm-specific context, $\mathcal{D}$ deterministically parses the LLM output, and $\mathcal{A}$ maps the decoded control signal and current configuration to the next configuration, optionally using search progress. We consider zero-shot policies with no controller-specific fine-tuning, grounded by quantitative state observations and recent interaction history.

This formulation separates two control interfaces. In **direct-mapping policies**, $\mathbf{c}^{(t)} \in \Lambda$ is itself an absolute configuration and $\mathcal{A}(\mathbf{c}^{(t)}, \boldsymbol{\lambda}^{(t)}, t/T) = \mathbf{c}^{(t)}$. In **ordinal-assessment policies**, $\mathbf{c}^{(t)}$ contains bounded control scores and $\mathcal{A}$ performs deterministic numerical actuation. HypaEvo instantiates the latter interface, as detailed in Section 4.

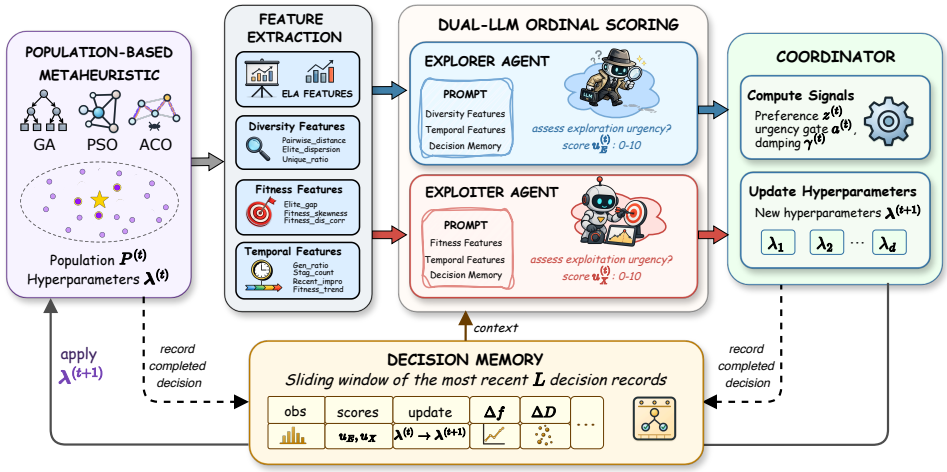


Figure 1: **Overview of HypaEvo.** At each decision point, structured search-state features are partitioned into diversity, fitness-quality, and temporal groups. Two specialized LLM assessors, the Explorer and the Exploiter, independently score exploration and exploitation urgency on a 0–10 ordinal scale. A deterministic coordinator then converts the score differential into bounded, range-normalized hyperparameter updates via a fixed direction map, with parameter bounds and directional priors enforced by construction. A sliding-window memory of recent actions and outcomes is appended to both assessor prompts.

4 METHODOLOGY

We introduce **HypaEvo** (Figure 1), a capability-aligned architecture that separates *semantic assessment* from *numerical actuation*: two specialized LLM assessors produce bounded ordinal urgency scores, and a deterministic coordinator maps these scores into constrained local updates. This operationalizes the capability-aligned principle from Section 1 along three axes: (i) LLMs perform bounded ordinal assessment in their most reliable regime; (ii) the search state is split into diversity and fitness-quality views evaluated by separate agents, exposing the exploration–exploitation trade-off as inspectable signals; and (iii) the score-to-parameter conversion is handled by deterministic code, with bounds and directional priors enforced by construction.

4.1 ARCHITECTURE OVERVIEW

HypaEvo operates at discrete decision points during the execution of a population-based metaheuristic, with decision interval Δ. Between decision points the metaheuristic evolves normally; at each decision point, HypaEvo computes structured features, obtains two LLM-generated urgency scores, and updates the hyperparameters for the next interval. The observation is partitioned as $\mathbf{o}^{(t)} = (\mathbf{o}_{\mathrm{D}}^{(t)}, \mathbf{o}_{\mathrm{F}}^{(t)}, \mathbf{o}_{\mathrm{T}}^{(t)})$ into diversity, fitness-quality, and temporal-progress groups, respectively. The control loop is

$$u_E^{(t)} = \mathcal{D}_{\mathrm{score}}\big(\mathcal{L}_{\mathrm{E}}(\textsc{Prompt}_{\mathrm{E}}(\mathbf{o}_{\mathrm{D}}^{(t)}, \mathbf{o}_{\mathrm{T}}^{(t)}, \mathcal{H}^{(t)}))\big) \in \{0, \ldots, 10\}, \tag{2}$$

$$u_X^{(t)} = \mathcal{D}_{\mathrm{score}}\big(\mathcal{L}_{\mathrm{X}}(\textsc{Prompt}_{\mathrm{X}}(\mathbf{o}_{\mathrm{F}}^{(t)}, \mathbf{o}_{\mathrm{T}}^{(t)}, \mathcal{H}^{(t)}))\big) \in \{0, \ldots, 10\}, \tag{3}$$

$$\boldsymbol{\lambda}^{(t+1)} = \texttt{Coord}\big(u_E^{(t)}, u_X^{(t)}, \boldsymbol{\lambda}^{(t)}, t/T\big), \tag{4}$$

where $\mathcal{L}_{\mathrm{E}}$ (Explorer) and $\mathcal{L}_{\mathrm{X}}$ (Exploiter) are the two LLM assessors producing urgency scores $u_E^{(t)}$ and $u_X^{(t)}$, $\mathcal{D}_{\mathrm{score}}$ is a deterministic parser that instantiates the generic decoder $\mathcal{D}$ from Eq. (1), specialized to bounded ordinal scores, `Coord` is the deterministic coordinator detailed in Section 4.4, and $\mathcal{H}^{(t)}$ is a sliding-window decision memory of length L retaining the most recent entries of the full history; the choice of L is studied empirically in Section 5. The normalized progress signal t/T enables late-stage damping in the coordinator. The architecture requires *exactly two* LLM calls per decision point regardless of how many hyperparameters are controlled.

4.2 FEATURE TAXONOMY

HypaEvo grounds LLM reasoning in three feature groups: diversity, fitness-quality, and temporal progress (Table 1). This three-way partition operationalizes the dimensional decomposition principle introduced in Section 4.1: diversity features are routed exclusively to the Explorer, fitness-quality features exclusively to the Exploiter, and temporal-progress features are shared across both assessors as a common reference frame. Features are presented in their natural scale: ratios and progress indicators bounded in $[0, 1]$ by construction (e.g., the unique solution ratio u, the fitness-distance correlation $|\mathrm{FDC}|$, the generation ratio ρ_t) are passed directly, while statistical features (skewness, elite gap) are presented in their conventional units. Counts and integer indicators are passed as-is. Exact definitions are given in Appendix C.

Table 1: Feature taxonomy: the Explorer receives diversity and temporal features; the Exploiter receives fitness and temporal features.

Feature	Description
Diversity features $\mathbf{o}_{\mathrm{D}}^{(t)}$	
Unique solution ratio u	Fraction of distinct individuals.
Mean pairwise distance $\bar{d}$	Average pairwise distance under a task-specific metric.
Elite dispersion ratio D_{elite}	Elite dispersion relative to the whole population.
Fitness-quality features $\mathbf{o}_{\mathrm{F}}^{(t)}$	
Fitness-distance correlation $\lvert\mathrm{FDC}\rvert$	Correlation between fitness and distance to the current best.
Elite gap g	Normalized gap between mean and best fitness.
Fitness skewness S	Skewness of the population fitness distribution.
Temporal features $\mathbf{o}_{\mathrm{T}}^{(t)}$	
Generation ratio ρ_t	Fraction of computational budget consumed.
Stagnation count σ	Generations without best-so-far improvement.
Recent improvement Δ_{best}	Best-so-far improvement over a trailing window.
Variability V	Ratio of trailing-window to current mean fitness.

4.3 Dual-LLM Ordinal Scoring

Each assessor receives a structured prompt containing its designated feature group, the shared temporal context, and the decision memory, and returns a single integer score on the ordinal scale $\{0, \ldots, 10\}$ that is parsed by $\mathcal{D}_{\mathrm{score}}$. The Explorer rates how urgently the search requires additional diversity, while the Exploiter rates how urgently it should intensify around promising regions. To make the scale reproducible across calls, both prompts use a shared semantic rubric with five anchors, namely *low* (0–2), *mild* (3–4), *moderate* (5–6), *high* (7–8), and *critical* (9–10), and instruct each assessor to judge urgency relative to the current search stage rather than against an absolute reference. The scores are not interpreted as calibrated value-function estimates; they need only be comparable as ordinal urgency levels under the shared rubric, which is sufficient for the coordinator to determine the relative pressure toward exploration or exploitation.

This scoring formulation is the central interface change introduced by HypaEvo. It replaces the final actuation step of direct-mapping controllers, namely the emission of absolute parameter values, with a bounded ordinal judgment aligned with the LLM's most reliable mode. The two-assessor split further routes diversity and fitness-quality features to specialized agents under a shared temporal context, exposing the exploration–exploitation trade-off as inspectable signals rather than collapsing it inside a single numerical output.

4.4 Deterministic Coordinator

The coordinator converts the two urgency scores into a valid local hyperparameter update. Its purpose is not to learn an optimal policy, but to provide a stable, interpretable actuation layer with structural guarantees by construction.

Let $z^{(t)} = (u_E^{(t)} - u_X^{(t)})/10 \in [-1, 1]$ be the normalized exploration–exploitation preference (positive favors exploration, negative favors exploitation, near-zero indicates no clear direction). Let $a^{(t)} = \min(1, (u_E^{(t)} + u_X^{(t)})/\tau_{\mathrm{ref}})$ be an urgency gate (with $\tau_{\mathrm{ref}} = 10$) that suppresses unnecessary movement when both scores are low. With normalized progress $\rho_t = t/T$, the late-stage damping factor is

$$\gamma(\rho_t) = \begin{cases} 1, & \rho_t \le \rho_0, \\ 1 - \beta \dfrac{\rho_t - \rho_0}{1 - \rho_0}, & \rho_t > \rho_0, \end{cases}$$

where $\rho_0 \in [0, 1)$ is the damping start and $\beta \in [0, 1]$ controls the maximum damping strength, so γ decreases linearly from 1 at ρ_0 to $1 - \beta$ at the end of the run. For each controlled hyperparameter

$j \in \{1, \ldots, d\}$, λ_j is associated with a directional prior $d_j \in \{+1, -1\}$, where $d_j = +1$ indicates that increasing λ_j biases the algorithm toward exploration (and conversely for $d_j = -1$). With $R_j = \lambda_j^{\max} - \lambda_j^{\min}$ denoting the parameter range width, the update rule is

$$\lambda_j^{(t+1)} = \Pi_{\Lambda_j}\left(\lambda_j^{(t)} + \eta_0 \cdot R_j \cdot d_j \cdot z^{(t)} \cdot a^{(t)} \cdot \gamma(t/T)\right), \tag{5}$$

where $\eta_0 > 0$ is a dimensionless base step size and Π_{Λ_j} projects the value back onto the valid domain. Integer-valued parameters are rounded to the nearest valid integer; categorical parameters require a separate discrete action map and are left for future extension.

Coordinator properties. The update rule provides three structural guarantees by construction. *Feasibility* follows from the projection Π_{Λ_j}, which ensures $\lambda_j^{(t+1)} \in \Lambda_j$ regardless of LLM output. *Bounded magnitude* follows from $|z^{(t)}|, a^{(t)}, \gamma(\rho_t) \in [0, 1]$, which yields $|\lambda_j^{(t+1)} - \lambda_j^{(t)}| \leq \eta_0 R_j$ before clipping. *Directional consistency* follows from the sign factor $d_j \cdot z^{(t)}$, which forces every update to respect the directional prior of λ_j. When both scores are high but balanced ($u_E = u_X$), $z^{(t)} = 0$ and no unilateral movement occurs, allowing subsequent observations to disambiguate. As a worked example, in a genetic algorithm with $d_{p_m} = +1$ and $d_{p_c} = -1$, an Explorer score exceeding the Exploiter score increases p_m and decreases p_c, promoting exploration.

4.5 Decision Memory and Algorithm

A sliding-window decision memory $\mathcal{H}^{(t)}$ retains the L most recent decision records. A record is created when a control decision is made and its outcome fields are completed at the next decision point, after the corresponding search interval has elapsed. We summarize these outcomes by Δf, the change in best-so-far fitness, and Δ_D, the change in mean pairwise distance. Recording both prevents the memory from biasing toward exploitation alone, since exploration may not improve best-so-far fitness immediately but can still register as a positive diversity change. The memory is formatted as a chronological text summary and appended to both assessor prompts, providing a compact structured record of recent decisions rather than an unrestricted chat history. We use $L = 20$ by default and evaluate sensitivity in the ablation study.

The full procedure is summarized in Algorithm 1 (Appendix D). HypaEvo extends to other metaheuristics by specifying the corresponding feature extractor and direction map while retaining the scoring and coordination interfaces; the GA, PSO, and ACO instantiations are detailed in Appendix E.

5 Experiments and Results

We evaluate HypaEvo along four questions. **RQ1** Does bounded ordinal scoring improve online hyperparameter control compared with direct LLM numerical generation? **RQ2** Where do the gains come from: state-dependent LLM assessment, dual-agent decomposition, or deterministic coordination? **RQ3** Does the method transfer across problem classes and metaheuristic families? **RQ4** How efficient and robust is HypaEvo across inference cost, LLM backbones, and prompt variants?

Benchmarks and Baselines. The main-text evaluation focuses on TSP and CVRP, while additional FSSP and UAV trajectory-optimization experiments, together with PSO results, are reported in Appendix A. We compare against three categories of baselines. Unless otherwise stated, baseline implementations are based on the authors' released code and are re-run by us under the matched experimental settings used in this study. *(1) Automatic hyperparameter tuning methods*: PT (Joshi & Bansal, 2020), a manually designed tuning scheme; GLEET (Ma et al., 2024), the SOTA reinforcement learning-based tuner, which we retrained strictly following the original settings for each algorithm and dataset; and BEA (Lan et al., 2022), a hybrid Bayesian-Evolutionary tuner. *(2) Neural combinatorial optimization*: DACT (Ma et al., 2021) and LEHD (Luo et al., 2023), two advanced learning-based solvers. *(3) LLM-enhanced optimization methods*: ReEvo (Ye et al., 2024) and EoH (Liu et al., 2024a), SOTA methods that leverage LLMs to enhance metaheuristic operators; and AutoEP (Xu et al., 2026), the closest baseline, which also uses LLMs for metaheuristic hyperparameter control but relies on direct numerical generation rather than bounded ordinal scoring.

Table 2: Comparison with various baselines on TSP under the GA backbone. Gap represents the percentage gap between the average run result and the optimal solution; a smaller value is better. Time is the average runtime (unit: minute).

Methods	eil51		Rd100		Kroa150		rd400		rat575		dsj1000	
	Gap(%)↓	*Time*	*Gap(%)*↓	*Time*	*Gap(%)*↓	*Time*	*Gap(%)*↓	*Time*	*Gap(%)*↓	*Time*	*Gap(%)*↓	*Time*
DACT	0.03%	0.5(m)	0.11%	4.5(m)	0.14%	7.3(m)	1.65%	17.1(m)	2.51%	24.8(m)	5.01%	58.5(m)
LEHD	0.09%	0.2(m)	0.26%	0.3(m)	0.92%	0.3(m)	2.01%	0.8(m)	2.67%	0.9(m)	5.39%	1.6(m)
GA	1.78%	0.4(m)	4.25%	0.5(m)	5.18%	0.6(m)	6.57%	0.9(m)	9.29%	1.0(m)	15.58%	1.5(m)
GA+PT	1.39%	0.5(m)	3.51%	0.7(m)	4.62%	0.8(m)	6.18%	1.2(m)	8.16%	1.2(m)	15.07%	1.9(m)
GA+GLEET	1.34%	0.9(m)	3.46%	1.2(m)	4.45%	1.4(m)	6.14%	1.7(m)	7.99%	2.6(m)	14.61%	3.6(m)
GA+BEA	1.38%	0.7(m)	3.62%	0.9(m)	4.59%	1.1(m)	6.22%	1.4(m)	8.15%	1.5(m)	14.83%	2.4(m)
GA+EoH	0.98%	0.5(m)	2.95%	0.6(m)	4.41%	0.7(m)	6.71%	0.9(m)	8.09%	1.1(m)	15.38%	1.6(m)
GA+ReEvo	0.95%	0.5(m)	2.90%	0.8(m)	4.27%	0.8(m)	6.59%	0.9(m)	8.21%	1.2(m)	14.94%	1.8(m)
GA+AutoEP	1.36%	7.3(m)	3.34%	7.7(m)	4.37%	8.1(m)	5.86%	10.4(m)	7.32%	11.9(m)	13.61%	16.9(m)
GA+HypaEvo (ours)	**0.94%**	**4.6(m)**	**2.89%**	**4.9(m)**	**3.91%**	**5.2(m)**	**5.19%**	**7.7(m)**	**6.53%**	**9.1(m)**	**11.38%**	**13.3(m)**
GA-2opt	0.21%	0.6(m)	0.61%	0.6(m)	0.87%	0.7(m)	1.92%	0.9(m)	2.95%	1.2(m)	4.01%	3.2(m)
GA-2opt+PT	0.18%	0.7(m)	0.49%	0.8(m)	0.52%	1.1(m)	1.51%	1.2(m)	2.41%	1.7(m)	3.23%	4.5(m)
GA-2opt+GLEET	0.14%	0.9(m)	0.40%	1.0(m)	0.41%	1.5(m)	1.48%	1.6(m)	2.38%	2.9(m)	2.98%	5.8(m)
GA-2opt+BEA	0.16%	0.8(m)	0.45%	0.9(m)	0.48%	1.2(m)	1.45%	1.5(m)	2.42%	2.1(m)	3.12%	5.1(m)
GA-2opt+EoH	0.11%	0.6(m)	0.21%	0.6(m)	0.24%	0.7(m)	1.61%	1.1(m)	2.49%	1.4(m)	3.22%	3.5(m)
GA-2opt+ReEvo	0.12%	0.6(m)	0.19%	0.7(m)	0.26%	0.9(m)	1.59%	1.3(m)	2.51%	1.3(m)	3.19%	3.7(m)
GA-2opt+AutoEP	0.11%	9.9(m)	0.12%	9.9(m)	0.09%	10.1(m)	0.87%	11.4(m)	2.13%	12.2(m)	2.97%	17.7(m)
GA-2opt+HypaEvo (ours)	**0.08%**	**5.0(m)**	**0.00%**	**5.4(m)**	**0.00%**	**5.5(m)**	**0.54%**	**8.2(m)**	**1.64%**	**9.7(m)**	**2.15%**	**15.6(m)**

LLM configuration and Protocol. Unless otherwise stated, LLM-based comparisons use the same backbone and decoding settings. Matched ablations additionally share the same optimization budget and decision schedule, while each method retains its native prompting and control architecture. HypaEvo uses two specialized LLM assessors that output bounded ordinal scores, which are mapped to hyperparameter updates by a lightweight coordinator. Prompts and coordinator details are provided in Appendix F and Appendix D. We report the mean optimality gap (%) when best-known solutions are available, runtime, LLM calls, and token usage. Detailed evaluation protocol, implementation settings, and base solver configurations are provided in Appendix D.1.

5.1 Main Results

TSP and CVRP. Table 2 reports the main TSP results on six TSPLIB instances. HypaEvo improves over AutoEP under both GA backbones: with plain GA, the average gap decreases from 5.98% to 5.14% while runtime drops from 10.38 to 7.47 minutes; with GA-2opt, the average gap decreases from 1.05% to 0.74%, reaching the optimum on rd100 and KroA150. The advantage persists on larger instances, reducing the gap from 0.87% to 0.54% on rd400, 2.13% to 1.64% on rat575, and 2.97% to 2.15% on dsj1000. These results show that the ordinal-scoring controller remains effective even when the underlying GA is strengthened by local search.

Figure 2 reports convergence under the ACO backbone on four CVRP scales. HypaEvo achieves the lowest final objective value on all four instances. On CVRP100, HypaEvo and AutoEP are comparable in the final regime, whereas from CVRP200 onward HypaEvo continues to improve after AutoEP begins to plateau; the separation is most pronounced on CVRP300 and CVRP500. The slower early adaptation but sustained later improvement is consistent with HypaEvo's bounded, gradual control updates and extends the evidence beyond the GA-based TSP setting.

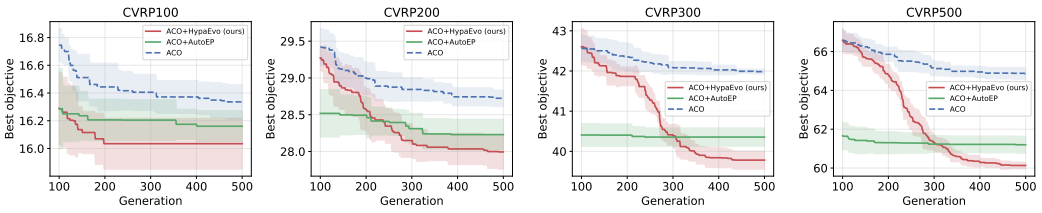


Figure 2: CVRP convergence curves under the ACO backbone. Curves show the best objective value over generations for ACO, ACO+AutoEP, and ACO+HypaEvo on CVRP100, CVRP200, CVRP300, and CVRP500. Shaded regions indicate variability across independent runs. HypaEvo reaches lower final objective values and sustains improvement later in the run, particularly on the larger instances.

FSSP and UAV trajectory optimization. Results on these additional problem classes, together with validation using the PSO metaheuristic, are reported in Appendix A.

5.2 Ablation Studies

LLM Scoring vs. the Deterministic Coordinator

A central attribution question is whether HypaEvo's gains arise from the LLM-based state assessment or primarily from the handcrafted coordinator. To isolate these roles, we keep the GA backbone, initialization, parameter bounds, direction map, urgency factor, damping schedule, update frequency, search budget, and coordinator fixed, and replace only the source of the two urgency scores (u_E, u_X). Figure 3 summarizes the resulting optimality gaps across instances and their average.

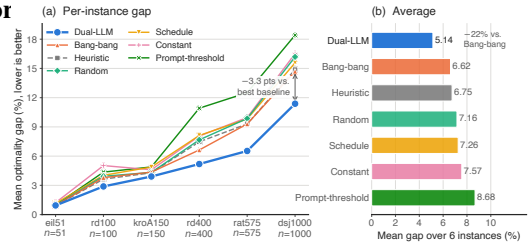


Figure 3: Same-coordinator scorer replacement on TSP. Per-instance and average optimality gap (%) under a fixed coordinator; lower is better.

Dual-LLM scoring is best on five of the six instances and achieves an average gap of 5.14, compared with 6.62 for the strongest non-LLM control (Bang-bang), a 22.3% relative reduction. Since the coordinator and optimization protocol are fixed across all variants, the result indicates that the coordinator alone does not explain the gain: state-dependent LLM assessment provides additional information about when exploration or exploitation is required. Detailed scorer definitions are provided in Appendix A.2.

Scoring vs. Direct Generation. To test whether ordinal scoring provides a more reliable interface than direct numerical generation, we compare four variants crossing *agent count* (single vs. dual) with *output format* (score vs. value). Table 3 shows that Dual-Score performs best across all evaluated instances, while Dual-Direct does not consistently improve over Single-Direct, so a second assessor alone is insufficient. Appendix A.1 further shows that Single-2Score, in which one assessor emits the same two urgency scores, underperforms Dual-Score and both baselines using a single assessor on all three appendix instances. This indicates that two outputs alone do not resolve the trade-off. With two assessors fixed, the proposed feature partition also outperforms full overlap and reversed assignment. Together, these results support specialized assessment, evidence routing by role, and deterministic actuation.

Table 3: Scoring format × agent structure. Mean optimality gap (%) over 30 runs. Single-LLM variants use the full feature set, while dual-LLM variants use the partition in Section 4. **Bold** indicates the best result per column.

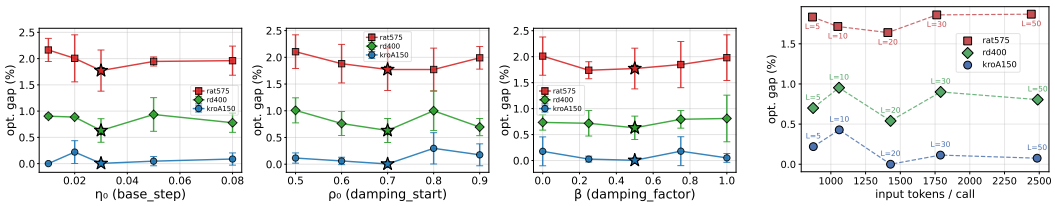
Variant	Description	rd100	KroA150	rd400	rat575	dsj1000
Single-Direct	Single LLM, direct control	0.15%	0.21%	1.52%	2.11%	2.98%
Single-Score	Single LLM, score-based control	0.65%	0.09%	1.23%	1.95%	2.64%
Dual-Direct	Dual LLMs, direct control	1.34%	0.09%	0.91%	1.94%	3.57%
Dual-Score (proposed)	**Dual LLMs, score-based control**	**0.00%**	**0.00%**	**0.54%**	**1.64%**	**2.15%**

Feature Partition, Coordinator Design, and Score Scale Resolution. These design choices are further evaluated against alternative variants, with the proposed configurations consistently achieving the best performance; full results are provided in Appendix A.1.

5.3 Sensitivity Analysis

We evaluate robustness to the coordinator's hyperparameters by varying one parameter at a time, with 30 runs for each setting. Figure 4a reports the mean optimality gap as a function of the base step size η_0, damping start ρ_0, and damping factor β, with the default value marked by a star.

The results show that HypaEvo is reasonably stable around the default configuration. Among the three hyperparameters, η_0 has the clearest effect: too small a step makes the controller conservative, while too large a step can overreact to transient score differences. The damping-related parameters ρ_0 and β have smoother effects, suggesting that HypaEvo does not rely on a finely tuned damping schedule. Overall, the default setting $(\eta_0, \rho_0, \beta) = (0.03, 0.7, 0.5)$ lies in a stable region across all tested instances, supporting the robustness of the deterministic coordinator.



(a) Coordinator hyperparameter sensitivity (b) Memory size effect

Figure 4: Sensitivity analysis on TSP. (a) Markers/error bars show mean ± standard deviation across 30 runs; ⋆ denotes the default value. (b) Memory-size trade-off between input tokens per call and optimality gap.

Memory size. Figure 4b shows a clear context–performance trade-off. A small memory provides limited historical context, whereas increasing the window beyond $L = 20$ raises input-token cost without consistent quality gains. The default $L = 20$ therefore offers the best balance across the three tested instances, with the lowest or near-lowest gaps at moderate context cost.

5.4 Backbone Sensitivity and Inference Cost

To examine whether HypaEvo is sensitive to the choice of LLM backbone, we compare it with AutoEP on rat575 using both open-weight and proprietary API-based models. As shown in Table 4, HypaEvo achieves comparable or better solution quality across all tested backbones, while consistently reducing runtime and API cost. The main efficiency gain comes from the scoring interface. Figure 5 shows that HypaEvo uses fewer input tokens and substantially fewer output tokens than AutoEP. The reduction is especially pronounced for output tokens, because each HypaEvo assessor only returns a bounded ordinal score, whereas AutoEP requires more verbose numerical-control generation. Thus, HypaEvo is not only competitive in optimization quality, but also more efficient as an LLM-driven controller.

Table 4: Backbone sensitivity and inference cost on rat575 over 10 runs. Optimization gap and runtime are reported as mean ± std. API cost is reported only for OpenAI models; Qwen3-30B is run locally.

Backbone	Method	*Gap (%)*	*Time (s)*	*Cost*
Qwen3-30B	**HypaEvo (ours)**	**1.72 ± 0.35%**	**398 ± 21**	0$
	AutoEP	1.90 ± 0.39%	609 ± 43	0$
GPT-4o-mini	**HypaEvo (ours)**	**2.11 ± 0.37%**	**1073 ± 70**	**0.74$**
	AutoEP	2.20 ± 0.40%	1317 ± 44	0.97$
GPT-4o	**HypaEvo (ours)**	**2.09 ± 0.37%**	**1114 ± 55**	**3.07$**
	AutoEP	2.10 ± 0.43%	1163 ± 98	4.05$

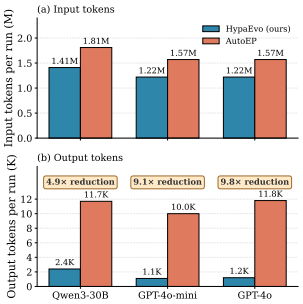


Figure 5: Token consumption.

6 Conclusion and Limitations

We introduced HypaEvo, a dual-LLM scoring architecture for dynamic hyperparameter control in population-based metaheuristics. Rather than generating precise numerical hyperparameters, HypaEvo uses LLMs as bounded ordinal assessors of exploration and exploitation urgency and delegates numerical actuation to a deterministic coordinator, enforcing parameter bounds and directional priors by construction. Across matched metaheuristic backbones, HypaEvo improves over AutoEP in solution quality while reducing inference overhead, and the ablations support the combined contribution of ordinal scoring, role specialization, and deterministic actuation. Current limitations include fixed direction maps, no direct support for categorical hyperparameters, and a manually designed feature taxonomy; future work can address these through state-dependent direction maps, discrete action mechanisms, and automated feature selection. Overall, the results support capability-aligned LLM interfaces that use models as assessors rather than numerical actuators for optimization control.

REFERENCES

Steven Adriaensen, André Biedenkapp, Gresa Shala, Noor Awad, Theresa Eimer, Marius Lindauer, and Frank Hutter. Automated dynamic algorithm configuration. *Journal of Artificial Intelligence Research*, 75:1633–1699, 2022. doi: 10.1613/jair.1.13922.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

André Biedenkapp, H. Furkan Bozkurt, Theresa Eimer, Frank Hutter, and Marius Lindauer. Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework. In *ECAI 2020 – 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 427–434. IOS Press, 2020. doi: 10.3233/FAIA200122.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Qiuyi Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32053–32068, 2022. doi: 10.52202/068431-2323.

Leonardo Lucio Custode, Fabio Caraffini, Anil Yaman, and Giovanni Iacca. An investigation on the use of Large Language Models for hyperparameter tuning in Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 1838–1845. ACM, 2024. doi: 10.1145/3638530.3664163.

Marco Dorigo, Mauro Birattari, and Thomas Stützle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006. doi: 10.1109/MCI.2006.329691.

A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999. doi: 10.1109/4235.771166.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023.

Leander Girrbach, Chi-Ping Su, Tankred Saanum, Richard Socher, Eric Schulz, and Zeynep Akata. Reference-free rating of LLM responses via latent information. *arXiv preprint arXiv:2509.24678*, 2025.

Charles Godfrey, Ping Nie, Natalia Ostapuk, David Ken, Shang Gao, and Souheil Inati. Likert or not: LLM absolute relevance judgments on fine-grained ordinal scales. *arXiv preprint arXiv:2505.19334*, 2025.

Hongshu Guo, Zeyuan Ma, Yining Ma, Xinglin Zhang, Wei-Neng Chen, and Yue-Jiao Gong. DesignX: Human-competitive algorithm designer for black-box optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.17866.

Susheel Kumar Joshi and Jagdish Chand Bansal. Parameter tuning for meta-heuristics. *Knowledge-Based Systems*, 189:105094, 2020.

Giorgos Karafotias, Mark Hoogendoorn, and A. E. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015. doi: 10.1109/TEVC.2014.2308294.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pp. 1942–1948. IEEE, 1995. doi: 10.1109/ICNN.1995.488968.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Gongjin Lan, Jakub M. Tomczak, Diederik M. Roijers, and A. E. Eiben. Time efficiency in optimization with a Bayesian-Evolutionary algorithm. *Swarm and Evolutionary Computation*, 69:100970, 2022. doi: 10.1016/j.swevo.2021.100970.

Robert Tjarko Lange, Yingtao Tian, and Yujin Tang. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*. ACM, 2024.

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 *International Conference on Machine Learning (ICML)*, pp. 32201–32223, 2024a. Oral presentation (Top 1.5%).

Haoqiang Liu, Zefang Zong, Yong Li, and Depeng Jin. NeuroCrossover: An intelligent genetic locus selection scheme for genetic algorithm using reinforcement learning. *Applied Soft Computing*, 146: 110680, 2023a.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2024b.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2511–2522. Association for Computational Linguistics, 2023b. doi: 10.18653/v1/2023.emnlp-main.153.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems*, volume 34, pp. 11096–11107, 2021.

Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Zhenrui Li, Guojun Peng, Yue-Jiao Gong, Yining Ma, and Zhiguang Cao. MetaBox: A benchmark platform for meta-black-box optimization with reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10775–10795, 2023. doi: 10.52202/075280-0474. Datasets and Benchmarks Track.

Zeyuan Ma, Jiacheng Chen, Hongshu Guo, Yining Ma, and Yue-Jiao Gong. Auto-configuring exploration-exploitation tradeoff in evolutionary computation via deep reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1497–1505. ACM, 2024.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 829–836. ACM, 2011. doi: 10.1145/2001576.2001690.

Sebastián Muñoz-Herrera and Karol Suchan. Constrained fitness landscape analysis of capacitated vehicle routing problems. *Entropy*, 24(1):53, 2022.

Gabriela Ochoa and Nadarajen Veerapen. Mapping the global structure of tsp fitness landscapes. *Journal of Heuristics*, 24(3):265–294, 2018.

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*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, volume 25, pp. 2951–2959, 2012.

Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1539–1546. ACM, 2005.

Tuhina Tripathi, Manya Wadhwa, Greg Durrett, and Scott Niekum. Pairwise or pointwise? Evaluating feedback protocols for bias in LLM-based evaluation. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2504.14716.

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.

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*, 2024. doi: 10.1109/TEVC.2024.3506731.

Zhenxing Xu, Yizhe Zhang, Weidong Bao, Hao Wang, Ming Chen, Haoran Ye, Wenzheng Jiang, Hui Yan, and Ji Wang. AutoEP: LLMs-driven automation of hyperparameter evolution for metaheuristic algorithms. In *International Conference on Learning Representations (ICLR)*, 2026. Oral presentation; arXiv:2509.23189.

Ke Xue, Jiacheng Xu, Lei Yuan, Miqing Li, Chao Qian, Zongzhang Zhang, and Yang Yu. Multi-agent dynamic algorithm configuration. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20147–20161, 2022. doi: 10.52202/068431-1465.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

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 (NeurIPS)*, 2024. arXiv:2402.01145.

Tuo Zhang, Jinyue Yuan, and Salman Avestimehr. Revisiting OPRO: The limitations of small-scale LLMs as optimizers. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1727–1735. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.100.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A ADDITIONAL RESULTS

A.1 ABLATIONS OF DESIGN COMPONENTS

This appendix presents the full ablation results summarized in the main text. We evaluate each design element by varying it in isolation while holding the remaining components fixed at their proposed configurations. All variants are evaluated on three TSP instances of increasing difficulty (KroA150, rd400, rat575) over thirty independent runs, with mean optimality gap reported in Table 5.

Table 5: Ablations of design components. Mean optimality gap (%) over 30 runs; lower is better. The proposed configuration is highlighted in each block.

Variant	Description	KroA150	rd400	rat575
Scoring and assessor decomposition				
Dual-Score (proposed)	**Two assessors, ordinal scoring**	**0.00%**	**0.54%**	**1.64%**
Single-Score	One assessor, one signed score	0.09%	1.23%	1.95%
Single-2Score	One assessor, two urgency scores	0.29%	1.92%	2.83%
Dual-Direct	Two assessors, direct numerical generation	0.09%	0.91%	1.94%
Single-Direct	One assessor, direct numerical generation	0.21%	1.52%	2.11%
Feature partition				
Proposed partition	**Role-aligned diversity/fitness split**	**0.00%**	**0.54%**	**1.64%**
Full overlap	Both agents observe all feature groups	0.12%	0.83%	1.82%
Reversed partition	Swap diversity and fitness groups across roles	0.21%	1.02%	2.19%
Coordinator design				
Proposed coordinator	**Gated and damped score-to-parameter mapping**	**0.00%**	**0.54%**	**1.64%**
No damping	Remove smoothing in parameter updates	0.00%	0.61%	1.79%
No urgency gate	Apply updates without urgency filtering	0.23%	1.26%	1.71%
LLM actuator	Let the LLM directly generate parameter values	0.10%	0.75%	2.32%
Score scale				
Proposed 11-level scale	**Scores in $\{0, \ldots, 10\}$**	**0.00%**	**0.54%**	**1.64%**
6-level scale	Scores in $\{0, 2, 4, 6, 8, 10\}$	0.16%	1.01%	1.98%
3-level scale	Scores in $\{0, 5, 10\}$	0.32%	1.53%	2.72%
21-level scale	Scores in $\{0, \ldots, 20\}$, rescaled to 0–10	0.43%	1.45%	2.02%

Scoring and Assessor Decomposition. We compare direct numerical generation with score-based control and additionally test whether separating exploration and exploitation across specialized assessors matters beyond the two-score representation itself. Dual-Score achieves the lowest mean gap on all three instances. In particular, Single-2Score asks one assessor to emit the same two urgency signals consumed by the deterministic coordinator, yet obtains gaps of 0.29%, 1.92%, and 2.83%, compared with 0.00%, 0.54%, and 1.64% for Dual-Score. Thus, merely representing exploration and exploitation with two scores is insufficient to explain the proposed method's advantage in these settings. Together with the feature-partition ablation below, the results support decomposing exploration and exploitation assessment into specialized, role-aligned assessors while retaining deterministic numerical actuation.

Feature Partition. We vary the feature partition while keeping the scoring format and coordinator fixed. The proposed role-aligned partition performs best, while both full-overlap and reversed-partition variants degrade performance. Giving both agents all features reintroduces signal conflation, while reversing the feature–role assignment breaks the intended exploration–exploitation specialization. The benefit of dimensional decomposition therefore comes from aligning each assessor with the evidence most relevant to its role, not merely from reducing context length.

Coordinator Design. The deterministic coordinator implements the separation between qualitative assessment and numerical actuation. Replacing it with an LLM actuator worsens performance, showing that ordinal scores alone are insufficient when the final update still relies on open-ended

numerical generation. Removing the urgency gate or late-stage damping also degrades results, indicating that stable control requires both suppressing unnecessary updates and shrinking disruptive changes near convergence. Deterministic actuation is therefore a core part of HypaEvo rather than an implementation detail.

Score Scale Resolution. We vary the ordinal scale to examine whether the coordinator benefits from magnitude information beyond mere directional preference. The proposed 11-level scale performs best, with the 3-level scale too coarse to convey magnitude and the 21-level scale showing degraded performance despite finer resolution. The pattern is consistent with the LLM-as-judge literature: finer scales increase the cognitive load on the assessor without proportionate gain in actionable information once the coordinator is configured to work with normalized differentials. The 11-level scale provides a bounded ordinal interface that is expressive enough for proportional control but not so fine-grained that it becomes another form of numerical estimation.

A.2 Same-Coordinator Scorer Replacement

Figure 3 in the main text isolates the contribution of the score source while keeping the plain GA backbone, initialization protocol, parameter bounds, direction map, urgency factor, damping schedule, update frequency, search budget, and deterministic coordinator fixed. We provide the scorer definitions and additional interpretation below.

Scorer definitions. *Heuristic* deterministically combines normalized diversity, stagnation, recent-improvement, and convergence indicators with equal weights. *Prompt-threshold* directly transcribes the qualitative threshold and cross-agent calibration rules used in the Explorer/Exploiter prompts. *Random* samples both urgency scores independently and uniformly from $\{0, \ldots, 10\}$. *Constant* fixes $u_E = u_X = 5$, disabling coordinator updates and retaining the controller's initial parameters; it is therefore a same-initialization scorer control rather than the plain-GA baseline in Table 2. *Bang-bang* switches between maximum exploration and exploitation according to sustained stagnation. *Schedule* follows a state-independent early-exploration/late-exploitation trajectory.

Dual-LLM scoring is best on five of the six instances and is within 0.02 gap points of the best control on eil51. Its average gap is 5.14, compared with 6.62 for the strongest non-LLM control (Bang-bang), a 22.3% relative reduction. This result indicates that the fixed coordinator alone does not explain the observed gains: the state-dependent LLM assessments provide additional information about when exploration or exploitation is required. These controls are deliberately training-free; GLEET remains the separate learning-based controller reference rather than a same-coordinator learned-scorer ablation.

A.3 Additional Problem Classes: FSSP and UAV Trajectory Optimization

We additionally evaluate HypaEvo on the Flow Shop Scheduling Problem (FSSP) and UAV trajectory optimization. The ordinal scoring interface and deterministic coordinator are unchanged; only solver-dependent feature instantiation, parameter domains, and direction priors are specified for the corresponding backbone. For every reported FSSP and UAV scale, we use 10 distinct test instances and evaluate each instance over 10 independent solver runs, so each method–scale entry aggregates 100 runs. For FSSP, the evaluation uses all 10 Taillard instances in each reported job–machine class (50 instances in total). For UAV trajectory optimization, instances are generated from a predeclared seed set shared by all methods using the released AutoEP generator.

Table 6: Additional FSSP results. Opt.gap is the percentage gap between the average run result and the optimal solution. Lower is better.

Method	N20m10	N50m10	N100m20	N200m20	N500m20	Average
GA-2opt	4.21%	4.98%	5.89%	5.52%	6.59%	5.438%
GA-2opt+PT	3.11%	3.71%	4.38%	3.94%	4.23%	3.874%
GA-2opt+GLEET	2.64%	2.96%	3.69%	3.23%	3.68%	3.240%
GA-2opt+BEA	2.89%	3.36%	3.92%	3.44%	3.79%	3.480%
GA-2opt+AutoEP	2.35%	2.83%	3.01%	3.19%	2.94%	2.864%
GA-2opt+HypaEvo	**1.95%**	**2.10%**	**2.69%**	**2.81%**	**2.72%**	**2.454%**

HypaEvo obtains the lowest reported gap on all five FSSP settings and reduces the average gap from 2.864 for AutoEP to 2.454, corresponding to a 14.3% relative reduction.

Table 7: Additional UAV trajectory-optimization results. Traj.Length is the drone flight-trajectory length; lower is better.

Method	N20↓	N50↓	N100↓	N200↓	N300↓	Average↓
ACO	157.29	312.27	613.37	1395.21	1915.25	878.678
ACO+PT	133.41	306.99	574.22	1187.56	1710.13	782.462
ACO+GLEET	140.97	305.16	565.53	1129.39	1690.05	766.220
ACO+BEA	130.89	297.84	569.64	1155.87	1703.16	771.480
ACO+AutoEP	121.32	289.63	553.83	1085.30	1576.66	725.348
ACO+HypaEvo	**120.71**	**283.72**	**549.07**	**1073.11**	**1571.87**	**719.696**

HypaEvo also obtains the lowest reported trajectory length at all five UAV scales. The average improvement over AutoEP is more modest, from 725.348 to 719.696 (0.78%), but is consistent across the evaluated scales. Together with TSP and CVRP, these results broaden the empirical evidence beyond routing while still leaving the feature instantiation, parameter domains, and direction priors solver-dependent.

A.4 CVRP WITH HGS AS AN EXTERNAL SOLVER REFERENCE

To contextualize absolute CVRP solution quality, we additionally report percentage gaps relative to HGS for two 2-opt-enhanced backbones. HGS is used only as an external specialized-solver reference; its gap is therefore 0% by construction. The matched comparisons of interest are AutoEP versus HypaEvo under the same GA-2opt or PSO-2opt backbone.

Table 8: CVRP results reported as percentage gaps relative to HGS; lower is better. HGS is the reference point of the metric, while the controller comparisons hold the underlying backbone fixed.

Method	Comparison role	CVRP100	CVRP200	CVRP300	CVRP500	Average
HGS	Specialized solver reference	0.00%	0.00%	0.00%	0.00%	0.000%
PSO-2opt	Base metaheuristic	2.11%	4.82%	6.20%	8.54%	5.418%
PSO-2opt+AutoEP	Matched controller	0.92%	2.91%	4.28%	5.15%	3.315%
PSO-2opt+HypaEvo	Matched controller	**0.85%**	**2.34%**	**3.78%**	**4.99%**	**2.990%**
GA-2opt	Base metaheuristic	1.65%	3.02%	4.54%	6.76%	3.993%
GA-2opt+AutoEP	Matched controller	0.75%	1.98%	2.15%	3.59%	2.118%
GA-2opt+HypaEvo	Matched controller	**0.54%**	**1.12%**	**1.63%**	**2.84%**	**1.533%**

HGS remains substantially stronger in absolute CVRP quality, as expected for a solver engineered specifically for CVRP. Under matched controller comparisons, HypaEvo improves over AutoEP on all eight backbone–scale comparisons. Averaged across the four scales, the HGS-relative gap decreases from 3.315 to 2.990 for PSO-2opt (9.8% relative reduction) and from 2.118 to 1.533 for GA-2opt (27.6% relative reduction).

A.5 PROMPT SENSITIVITY

We evaluate prompt sensitivity on the same GA-2opt backbone across TSP, CVRP, and FSSP while keeping the LLM backbone, state information, output schema, coordinator, optimization budget, seeds, and run counts matched across prompt variants. *Paraphrased* rewrites the role and instructions while preserving their semantics; *Reordered blocks* changes only the presentation order of the feature groups; and *Alternative anchor wording* rephrases the ordinal anchor descriptions. Each entry averages the six TSP instances, four CVRP scales, or five FSSP settings, respectively.

Table 9: Prompt sensitivity measured by mean optimality gap (%) under matched experimental settings; lower is better.

Prompt variant	TSP	CVRP	FSSP
Original	**0.735%**	1.533%	**2.454%**
Paraphrased	0.752%	**1.529%**	2.501%
Reordered blocks	0.757%	1.584%	2.625%
Alternative anchor wording	0.760%	1.543%	2.593%

The paraphrased prompt changes performance by $+2.3\%$ on TSP, -0.3% on CVRP, and $+1.9\%$ on FSSP. Reordering the feature blocks changes the corresponding results by $+3.0\%$, $+3.3\%$, and $+7.0\%$, while alternative anchor wording changes them by $+3.4\%$, $+0.7\%$, and $+5.7\%$. The largest observed degradation across these semantically equivalent variations is therefore 7.0%. These results support reasonable stability to ordinary prompt variations without implying complete prompt invariance.

A.6 Further Interpretation of the Empirical Results

What drives the gain? The controlled ablations provide complementary evidence about the source of HypaEvo's improvement. The same-coordinator experiment keeps the complete actuation rule fixed and changes only the source of the urgency scores; Dual-LLM scoring reduces the average gap from 6.62% for the strongest non-LLM control (Bang-bang) to 5.14%. The scoring-format ablation further compares score-based and direct numerical interfaces while holding the agent count fixed. Averaged over the five instances in Table 3, the gap changes from 1.39% to 1.31% in the single-agent setting and from 1.57% for Dual-Direct to 0.87% for Dual-Score, corresponding to a 44.8% relative reduction in the dual-agent comparison. The Direct variants emit structured numerical values that are validated and projected to the legal parameter domains, so the difference is not explained simply by malformed or out-of-range outputs. Taken together, these results support the view that the observed gains arise from the combination of state-dependent LLM assessment, role specialization, and deterministic actuation rather than from the coordinator alone.

Dual assessment and role specialization. To separate assessor decomposition from the two-score output representation, we add a Single-2Score variant in which one LLM produces both exploration and exploitation urgency scores, which are passed to the same deterministic coordinator used by Dual-Score. Single-2Score obtains mean gaps of 0.29%, 1.92%, and 2.83% on KroA150, rd400, and rat575, respectively, compared with 0.00%, 0.54%, and 1.64% for Dual-Score. Averaged across the three instances, Dual-Score reduces the gap from 1.68% to 0.73% (a 56.7% relative reduction). Notably, producing two urgency scores with a single assessor does not improve over the original Single-Score baseline on these instances, indicating that the benefit is not explained merely by representing exploration and exploitation with two numerical outputs. A single assessor must jointly interpret heterogeneous diversity, fitness-quality, and temporal signals while producing two competing judgments, which can blur the exploration–exploitation distinction. Dual-Score instead separates both the evidence streams and assessor roles, yielding more specialized control signals. The feature-partition ablation provides complementary evidence: with the two-assessor architecture, scoring format, and coordinator fixed, the proposed role-aligned partition outperforms both full feature overlap and reversed assignment on all three instances. Together, these results support decomposing exploration and exploitation assessment into specialized, role-aligned assessors in the evaluated settings.

Breadth and strength of the evidence. The additional experiments test whether the observed benefit is confined to the original TSP setting. On FSSP, HypaEvo reduces the average gap relative to AutoEP by 14.3%, whereas on UAV trajectory optimization the average improvement is smaller (0.78%) but remains consistent across all five evaluated scales. The HGS-relative CVRP results provide a complementary perspective: HGS remains substantially stronger in absolute solution quality, while HypaEvo consistently improves over AutoEP when the underlying GA-2opt or PSO-2opt backbone is held fixed. We therefore interpret these results as evidence that the scoring-and-

coordination interface transfers across the evaluated problem classes and metaheuristic families, rather than as a claim of universal solver independence.

Convergence and computational trade-off. The CVRP convergence curves and the endpoint comparisons capture different aspects of controller behavior. AutoEP often improves more rapidly early in the run, whereas HypaEvo adapts more gradually but continues improving after AutoEP begins to plateau, particularly on the larger CVRP instances. A plausible interpretation is that HypaEvo makes the control decision explicitly phase-aware. In addition to population-state features, both assessors observe temporal signals describing search progress, stagnation, recent best-so-far improvement, and population-level fitness variability. These signals help distinguish a search that is still making progress from one that has genuinely stagnated, and allow similar population states to be interpreted differently depending on the remaining search budget. Early in the run, maintaining sufficient exploration can therefore be preferable to aggressively pursuing short-horizon objective reduction; as the search matures, accumulated fitness structure and the reduced remaining budget can shift pressure toward exploitation. The deterministic coordinator further limits abrupt interventions through bounded, range-normalized updates and late-stage damping, yielding a more gradual adaptation trajectory. This behavior is aligned with the objective in Section 3.1, which targets the terminal best-so-far solution rather than early convergence speed. We therefore interpret the slower early improvement as consistent with search-horizon-aware control aimed at deeper final convergence, rather than as failed adaptation; the present experiments, however, do not establish a causal decomposition of these effects.

HypaEvo also incurs non-negligible LLM inference overhead relative to lightweight non-LLM controllers; its efficiency advantage is specifically relative to the directly comparable LLM-driven controller AutoEP. The practical benefit should therefore be interpreted as a quality–cost trade-off rather than zero control overhead: under the matched TSP experiments, HypaEvo improves both final quality and runtime relative to AutoEP, while local Qwen3-30B deployment further reduces latency and monetary cost.

B Extended Related Work

This appendix places HypaEvo within six related threads: dynamic algorithm configuration, fitness-landscape analysis, LLM-based optimization, deterministic execution, ordinal scoring, and alternative controller designs.

B.1 Dynamic Algorithm Configuration and Learned Online Control

Dynamic parameter adjustment predates LLM-based controllers. The taxonomy of Eiben et al. (1999) distinguishes deterministic, adaptive, and self-adaptive parameter control, while Karafotias et al. (2015) reviews later developments. Dynamic algorithm configuration (DAC) generalizes this setting by adapting parameters during execution instead of selecting one fixed configuration in advance (Biedenkapp et al., 2020; Adriaensen et al., 2022). HypaEvo belongs to this online setting, but derives its control signal from frozen LLM assessors rather than a controller trained across optimization instances.

Learned DAC often formulates configuration as reinforcement learning. GLEET (Ma et al., 2024) learns a state-to-action controller for the exploration and exploitation trade-off, while MetaBox provides a benchmark platform for MetaBBO-RL methods (Ma et al., 2023). MA-DAC assigns different configuration hyperparameters to cooperative RL agents (Xue et al., 2022). HypaEvo instead decomposes assessment: its agents separately evaluate exploration and exploitation evidence, while a shared deterministic coordinator performs numerical actuation. Thus, the agents specialize by assessment role rather than parameter ownership.

B.2 Fitness-Landscape Analysis and Online Search-State Descriptors

Fitness-landscape analysis characterizes the structure encountered by search algorithms. Prior work studies global TSP landscapes (Ochoa & Veerapen, 2018) and constrained CVRP landscapes (Muñoz-Herrera & Suchan, 2022). HypaEvo does not construct a complete landscape model. It repeatedly queries compact population and trajectory descriptors, including fitness-distance correlation, diversity,

elite dispersion, stagnation, and recent progress, to guide online control. Distance-based descriptors use representation-appropriate metrics; for routing problems, we use edge-based symmetric difference rather than positional Hamming distance. Richer landscape descriptors could be incorporated without changing the scoring or coordination interfaces.

B.3 LLMs in Evolutionary Computation and Hyperparameter Optimization

Three roles for LLMs in EC. Recent LLM4EC work can be organized by the role assigned to the model (Wu et al., 2024). As stand-alone optimizers, LLMs directly propose solutions from prior history, as in OPRO (Yang et al., 2024) and EvoLLM (Lange et al., 2024). Such methods can struggle with numerical generation and long optimization histories (Zhang et al., 2024; Xu et al., 2026). A second line uses LLMs to generate algorithm components offline, including EoH (Liu et al., 2024a), ReEvo (Ye et al., 2024), FunSearch (Romera-Paredes et al., 2024), and LLaMEA (van Stein & Bäck, 2025). A third line embeds LLMs inside an optimization loop to guide search or control algorithm behavior. HypaEvo and AutoEP belong to this third category.

Sequence-model and LLM interfaces for HPO. OptFormer represents heterogeneous HPO studies as text sequences and trains a Transformer to propose hyperparameters and predict objective responses (Chen et al., 2022). LLAMBO integrates LLMs into Bayesian optimization for warm-starting, surrogate modeling, and candidate sampling without LLM fine-tuning (Liu et al., 2024b). Both demonstrate language-based interfaces for HPO, but operate over black-box trials rather than control a running population-based metaheuristic from its evolving search state.

LLM-driven hyperparameter tuning before AutoEP. Custode et al. (2024) study LLM-based hyperparameter recommendation in evolutionary algorithms, including step-size adaptation in the $(1 + 1)$ Evolution Strategy. AutoEP (Xu et al., 2026) later grounds LLM reasoning in online ELA features and uses compositional agents for metaheuristic hyperparameter control. In both settings, the numerical parameter choice remains an LLM output. HypaEvo instead restricts the LLM to bounded urgency assessment and moves numerical updates to deterministic code.

Position of HypaEvo within LLM4EC. HypaEvo changes the interface rather than the placement of the LLM in the optimization loop. The model performs bounded assessment, while the coordinator handles arithmetic, projection, and directional consistency. This separation narrows the LLM's role and makes the actuation constraints explicit.

B.4 Delegating Numerical Computation to Deterministic Execution

A related design pattern appears in program-aided language models. PAL translates a reasoning problem into executable program steps and delegates computation to a deterministic runtime (Gao et al., 2023). PAL is not an optimization controller, but it provides a broader precedent for separating semantic reasoning from exact computation. HypaEvo applies this separation at the control boundary: LLMs assess search-state urgency, while code performs score conversion, parameter updates, projection, and enforcement of directional priors.

B.5 Ordinal Scoring as an LLM Interface

Rubric-conditioned judgment. Prometheus shows that evaluator language models can use explicit, fine-grained rubrics to produce criterion-specific assessments (Kim et al., 2024). HypaEvo does not train an evaluator or use reference answers, but similarly defines shared semantic anchors for its score levels. The score-resolution ablation then evaluates the chosen granularity empirically.

Pointwise versus pairwise. LLM-as-a-judge research has compared pointwise scoring with pairwise judgments. Earlier work (Zheng et al., 2023; Liu et al., 2023b) reports benefits of pairwise comparison for some biases, while later studies reveal trade-offs. Tripathi et al. (2025) find absolute pointwise scoring more robust to distractor-based evaluator manipulation, and Godfrey et al. (2025) show that sufficiently fine-grained pointwise scales can approach listwise ranking. These findings motivate HypaEvo's bounded pointwise interface, whose 11-level resolution is evaluated against coarser and finer alternatives.

Calibration and stability of ordinal scores. Ordinal scores can exhibit instability, compression, and ties across repeated LLM judgments (Girrbach et al., 2025). We do not adopt latent-information probes because they require model-internal access that is unavailable for API-only models. HypaEvo instead treats the scores as bounded control signals under a shared rubric rather than calibrated value estimates. Section 4.3 discusses this choice further.

Why not a forced binary choice between explore and exploit? A binary controller could ask the LLM to select *explore* or *exploit*, optionally with an intensity. We do not use this design because a binary choice discards the magnitude captured by $z^{(t)} \in [-1, 1]$ and may inherit known position and distractor biases (Tripathi et al., 2025). Independent pointwise scoring preserves graded pressure at the cost of one additional LLM call per decision point.

B.6 Design Alternatives Considered

For completeness, we briefly note three alternatives considered during the design of HypaEvo.

Single-agent ordinal scoring with a signed scale. A single LLM could emit one signed score $u \in \{-10, \ldots, +10\}$, with positive values favoring exploration and negative values favoring exploitation. This reduces the LLM-call budget to one per decision point, but forces one model to resolve diversity and fitness-quality evidence within the same inference. The Single-Score variant in Appendix A.1 evaluates this alternative.

LLM-based actuator with deterministic coordination context. Another design retains dual scoring but asks an LLM to map the two scores into hyperparameter values. This preserves the scoring stage but reintroduces numerical generation at the actuation boundary, losing the structural guarantees of Equation (5). The LLM actuator variant in Appendix A.1 evaluates this design.

Learned direction maps. The current direction map d_j is fixed from first-order operator priors. Learning it online from parameter-outcome records could increase flexibility, but would remove the simple directional-consistency guarantee and introduce an additional learning loop. We therefore keep the map fixed in this work and leave state-dependent or learned variants to future study.

C Feature Definitions

This appendix provides exact mathematical definitions for the features used by the Explorer and Exploiter assessors, matching the implementation. Let $X = (x_1, \ldots, x_n)$ denote the population at generation t, with each x_i a candidate solution, and let $y = (y_1, \ldots, y_n)$ be the corresponding fitness values under minimization. Let $x^\star$ denote the current best individual. For TSP, $E(x)$ denotes the edge set of tour x, and the edge-based distance between two tours is the size of the symmetric difference $|E(x_a) \triangle E(x_b)|$. When a feature is undefined, typically because a denominator or sample size falls below the listed threshold, it is replaced by a fixed sentinel value chosen to be recognizable as a boundary case in context.

C.1 Diversity Features (Explorer)

Unique solution ratio u. The fraction of distinct individuals in the population:

$$u = \frac{|\{x_1, \ldots, x_n\}|}{n} \in [0, 1].$$

For continuous representations, equality is determined under a small numerical tolerance. Lower values of u indicate genetic redundancy and potential premature convergence.

Mean pairwise distance $\bar{d}$. The average pairwise distance under the problem-specific metric (edge-based for TSP, see Section C.4):

$$\bar{d} = \frac{2}{n(n-1)} \sum_{i<j} d(x_i, x_j).$$

The quantity $\bar{d}$ provides a continuous-valued measure of population spread that complements the discrete-valued u.

Elite dispersion ratio D_{elite}. Let $q = \lceil 0.10\, n \rceil$, and let $\bar{d}_{\text{elite}}$ denote the mean pairwise distance among the q best-fitness individuals. The elite dispersion ratio is

$$D_{\text{elite}} = \frac{\bar{d}_{\text{elite}}}{\bar{d}} \in [0, 1+).$$

The ratio approaches 0 when elites have collapsed onto a small region while the population at large remains diverse, signalling exploitation pressure that may benefit from increased exploration. Values near or above 1 indicate that elites are spread comparably to the rest of the population. If $\bar{d} < 10^{-10}$, D_{elite} is set to 1.

C.2 Fitness-Quality Features (Exploiter)

Fitness-distance correlation $|\text{FDC}|$. Let $d_i = d(x_i, x^\star)$ denote the distance from individual i to the current best under the problem-specific metric. The fitness-distance correlation is the Pearson correlation between fitness and distance to the best:

$$\text{FDC} = \text{corr}(y, d), \qquad |\text{FDC}| \in [0, 1].$$

We use $|\text{FDC}|$ as a magnitude-of-association descriptor between fitness and distance to the current best, without interpreting the correlation sign itself as a direct indicator of exploitability. Larger magnitudes indicate a stronger linear association, whereas values near zero indicate weak linear association. We set $|\text{FDC}|$ to 0 if $\text{std}(d) < 10^{-10}$ or $\text{std}(y) < 10^{-10}$.

Elite gap g. The number of standard deviations separating the population mean from the best fitness:

$$g = \frac{\text{mean}(y) - \min(y)}{\text{std}(y)}.$$

Larger values of g indicate a more isolated elite, suggesting that the population has not yet caught up with the leading individual and that exploitation around the best may yield further gains. Set to 0 if $\text{std}(y) < 10^{-10}$.

Fitness skewness S. With $\mu = \text{mean}(y)$ and standard deviation $\text{std}(y)$ computed at the population level ($\text{ddof} = 0$),

$$S = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \mu}{\text{std}(y)} \right)^3.$$

Negative skewness indicates a left-skewed fitness distribution with most individuals clustered at higher (worse) fitness and a few standout elites, which is the typical signature of an exploitation-friendly state. Set to 0 if $\text{std}(y) < 10^{-12}$ or $n < 3$.

C.3 Temporal Features (Both Assessors)

Generation ratio ρ_t. The fraction of the computational budget consumed:

$$\rho_t = \frac{t+1}{T_{\max}} \in [0, 1].$$

Stagnation count σ_t. The number of consecutive generations since the most recent best-so-far improvement. For the minimization setting, with tolerance $\epsilon = 10^{-6}$,

$$\sigma_t = \max\Big\{ k \geq 0 : y^\star_{t-i} \leq y^\star_t + \epsilon,\ \forall i = 0, \ldots, k \Big\}.$$

Because the best-so-far sequence is non-increasing, $\sigma_t = 0$ immediately after a strict improvement and increases while the best value remains unchanged within tolerance.

Recent improvement Δ_{best} (window 5). The relative improvement of the best-so-far over the past five generations:

$$\Delta_{\text{best}} = \frac{y^\star_{t-5} - y^\star_t}{|y^\star_{t-5}|}.$$

In the prompt, Δ_{best} is presented as a percentage for human-readable interpretation, while the numerical value passed to the assessor is unchanged.

Variability V (window $|W| = 20$). Let W be the past 20 generations, or all available generations if fewer have elapsed. The variability is the ratio of trailing-window mean fitness to current mean fitness:

$$V = \frac{\frac{1}{|W|}\sum_{e \in W} \text{mean}(y_e)}{\text{mean}(y_t)}.$$

For minimization, $V > 1$ indicates that the population mean fitness has been higher in the past than at present, which signals an improving search trajectory.

C.4 Domain-Specific Distance Metric

The diversity features and the FDC depend on a distance function $d(\cdot, \cdot)$ between candidate solutions. For TSP, we use the edge-based symmetric-difference distance $|E(x_a) \triangle E(x_b)|$ rather than position-based Hamming distance. This choice corrects a measurement issue that arises with Hamming distance on tour representations: two tours can differ in every position while sharing nearly all edges, for instance under a cyclic rotation, which would otherwise inflate the apparent diversity and distort the FDC. The edge-based distance reflects the true structural similarity of tours and is invariant under rotational and directional reparameterizations. For other problems considered in this work, the distance metric is specified in the corresponding setup section.

D Algorithm and Implementation Details

Algorithm 1 summarizes the HypaEvo procedure referenced in Section 4.5. For clarity, the pseudocode uses a mutable buffer $\mathcal{H}$ as the implementation of the abstract history $\mathcal{H}^{(t)}$. At each decision point, HypaEvo observes the current search state, completes the outcome information for the preceding interval, obtains two urgency scores, selects the configuration for the next interval through the deterministic coordinator, and then advances the metaheuristic.

Algorithm 1 HypaEvo: Dual-LLM Ordinal Control for Dynamic Hyperparameters

Require: Metaheuristic $\mathcal{M}$, instance $\mathcal{I}$, horizon T, decision interval Δ, parameter domains $\{\Lambda_j\}_{j=1}^{d}$, direction map $\{d_j\}_{j=1}^{d}$, memory size L, coordinator hyperparameters $(\eta_0, \tau_{\mathrm{ref}}, \rho_0, \beta)$

Initialize latent state $s^{(0)} \leftarrow \textsc{InitState}(\mathcal{M}, \mathcal{I})$
Initialize hyperparameters $\boldsymbol{\lambda}^{(0)} \leftarrow \textsc{InitConfig}(\{\Lambda_j\}_{j=1}^{d})$
Initialize decision memory $\mathcal{H} \leftarrow \emptyset$ and feature buffer $\mathcal{B} \leftarrow \emptyset$
for $t = 0, 1, \ldots, T-1$ **do**
  $P^{(t)} \leftarrow \mathrm{Pop}(s^{(t)})$; append $\bar{f}(P^{(t)})$ to $\mathcal{B}$
  if $t \bmod \Delta = 0$ **then**
    if $\mathcal{H} \neq \emptyset$ **then**
      Complete the most recent pending record in $\mathcal{H}$ using outcomes observed since its decision time
    end if
    $(\mathbf{o}_{\mathrm{D}}^{(t)}, \mathbf{o}_{\mathrm{F}}^{(t)}, \mathbf{o}_{\mathrm{T}}^{(t)}) \leftarrow \textsc{ComputeFeatures}(P^{(t)}, \mathcal{B})$
    $u_E^{(t)} \leftarrow \mathcal{D}_{\mathrm{score}}\Big(\mathcal{L}_{\mathrm{E}}(\textsc{Prompt}_{\mathrm{E}}(\mathbf{o}_{\mathrm{D}}^{(t)}, \mathbf{o}_{\mathrm{T}}^{(t)}, \mathcal{H}))\Big)$
    $u_X^{(t)} \leftarrow \mathcal{D}_{\mathrm{score}}\Big(\mathcal{L}_{\mathrm{X}}(\textsc{Prompt}_{\mathrm{X}}(\mathbf{o}_{\mathrm{F}}^{(t)}, \mathbf{o}_{\mathrm{T}}^{(t)}, \mathcal{H}))\Big)$
    $\boldsymbol{\lambda}^{(t+1)} \leftarrow \texttt{Coord}(u_E^{(t)}, u_X^{(t)}, \boldsymbol{\lambda}^{(t)}, t/T)$
    Append a pending record $(\mathbf{o}^{(t)}, u_E^{(t)}, u_X^{(t)}, \boldsymbol{\lambda}^{(t)}, \boldsymbol{\lambda}^{(t+1)})$ to $\mathcal{H}$ and retain the most recent L records
  else
    $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)}$
  end if
  $s^{(t+1)} \leftarrow \Phi_{\mathcal{M}}\big(s^{(t)}, \boldsymbol{\lambda}^{(t+1)}; \xi^{(t)}\big)$
end for
return $x^\star = \arg\min_{x \in \bigcup_{t=0}^{T} \mathrm{Pop}(s^{(t)})} f_{\mathcal{I}}(x)$

Coordinator subroutine. The `Coord` step in Algorithm 1 implements Equation (5) from Section 4.4. Given the two urgency scores and the current configuration, it computes the score difference $z^{(t)} = (u_E^{(t)} - u_X^{(t)})/10$, the urgency gate $a^{(t)} = \min(1, (u_E^{(t)} + u_X^{(t)})/\tau_{\mathrm{ref}})$, the late-stage damping factor $\gamma(t/T)$, and applies the range-normalized update to each controlled hyperparameter independently. The full subroutine is implemented in fewer than thirty lines of code and requires no LLM inference.

Feature buffer. The auxiliary buffer $\mathcal{B}$ maintains a sliding window of the population mean fitness $\bar{f}(P^{(\tau)})$ over the past twenty generations to support computation of the variability feature V defined in Appendix C. Storing population summaries rather than full populations keeps the buffer cost $O(1)$ per generation independent of population size, and the window length is set to match the trailing window of V.

Feature computation cost. The ComputeFeatures step dominates the per-decision overhead outside LLM inference. The most expensive component is the mean pairwise distance, which is $O(N^2)$ in the population size. For all population sizes used in our experiments ($N \leq 500$), this is computed exactly; for larger populations, it can be estimated from a fixed-size random sample of pairs without affecting the rest of the pipeline. All other features are $O(N)$ or cheaper.

Prompt format. Each assessor prompt contains four blocks in order: a static role description identifying the assessor as Explorer or Exploiter, the assessor's feature group with feature-specific numerical values and validity flags, the shared temporal-progress features, and the decision memory formatted as a chronological text summary. The assessor is instructed to return a single integer in $\{0, \ldots, 10\}$ in a fixed JSON-like field, optionally followed by a free-text rationale used only for logging. Full prompt templates are provided in Appendix F.

D.1 Details of the Experimental Setup

All experiments are executed on our compute server equipped with an NVIDIA GeForce RTX 4090 GPU. Qwen3-30B is hosted locally on this server, whereas proprietary LLM backbones are queried through the official APIs of their providers. For comparison methods, we start from the authors' released implementations whenever available and re-run them under the matched evaluation protocol used in this study. Model-specific settings otherwise follow the corresponding original implementations unless explicitly stated.

AutoEP reproduction and comparability. The AutoEP results reported in this paper are obtained from our independent re-runs of the authors' released implementation under the matched protocol used for our comparisons; they are not copied from the values reported in the original AutoEP study (Xu et al., 2026). Within each matched comparison, AutoEP and HypaEvo use the same problem instance, underlying metaheuristic backbone, search budget, and random-seed set. The TSP instance suite also differs slightly from the published AutoEP evaluation: the original study reports `rd300`, whereas we use `rd400` as the corresponding intermediate-scale instance. Consequently, the absolute gaps and runtimes in our tables should not be compared one-to-one with the numbers reported in the original AutoEP paper, which were obtained under its original instance set, implementation configuration, and execution environment. All runtime values reported here are measured in our own environment described above, and the re-run AutoEP results are used consistently as the matched baseline throughout this paper.

For the original TSP/CVRP evaluation, each method–instance pair is run 30 times with independent random seeds. For FSSP and UAV trajectory optimization, each reported scale contains 10 test instances, with 10 independent runs per instance, yielding 100 runs per method–scale entry. Table 10 summarizes the common search budget and initialization settings used for the metaheuristic backbones. The listed hyperparameter values are the starting values before any online adjustment by HypaEvo.

Table 10: Search-budget and initialization settings for the metaheuristic backbones.

Algorithm	Parameter	Value
GA	Population size	500
	Maximum number of iterations	500
	Initial mutation probability	0.10
	Initial crossover probability	0.60
PSO	Swarm size	500
	Maximum number of iterations	500
	Initial inertia weight w	0.30
	Initial cognitive coefficient c_1	1.50
	Initial social coefficient c_2	1.50
ACO	Number of ants	500
	Maximum number of iterations	500
	Initial pheromone exponent α	1.00
	Initial heuristic exponent β	2.00
	Initial evaporation rate ρ	0.20

D.2 Code Availability and Reproducibility

To facilitate reproducibility during anonymous review, we include an anonymized code package in the supplementary material covering the core HypaEvo implementation, together with the implementation details in the appendix. Upon acceptance, we will publicly release the complete project repository, including all implementation code, experiment scripts, configuration files, prompts, project data, random seeds, evaluation utilities, and environment/setup instructions.

E METAHEURISTIC INSTANTIATIONS

HypaEvo is designed as an interface rather than a solver-specific controller: instantiating it for a new metaheuristic requires specifying only the feature extractor and the direction map, while the LLM scoring interface and deterministic coordinator (Sections 4.3, 4.4) remain unchanged. The direction maps below are not universal claims about each metaheuristic family; they are first-order priors for the operator variants used in our experiments and can be replaced when a different operator design changes the expected role of a parameter.

Genetic Algorithm. The controllable parameters are the mutation probability p_m and the crossover probability p_c, with ranges $p_m \in [0.10, 0.45]$ and $p_c \in [0.55, 0.70]$ following AutoEP (Xu et al., 2026). We use the direction map

$$d_{p_m} = +1, \qquad d_{p_c} = -1,$$

reflecting the standard heuristic that mutation pressure promotes diversity restoration (exploration) while crossover pressure promotes recombination of existing solution structures (exploitation under a converged or partially converged population) (Eiben et al., 1999). The features in Table 1 apply directly, with the diversity metric $\bar{d}$ computed using the edge-based symmetric difference for TSP tours.

Particle Swarm Optimization. The controllable parameters are the inertia weight w, the cognitive coefficient c_1, and the social coefficient c_2, with ranges $w \in [0.2, 1.0]$ and $c_1, c_2 \in [0.5, 2.5]$ following AutoEP (Xu et al., 2026). We use the direction map

$$d_w = +1, \qquad d_{c_1} = +1, \qquad d_{c_2} = -1,$$

reflecting the conventional role of inertia and cognitive attraction in preserving exploratory motion and of stronger social attraction in intensifying movement toward the global best (Kennedy & Eberhart, 1995). The scoring and coordination interfaces are unchanged; the representation-specific feature extractor and distance metric are instantiated for the evaluated PSO setting.

Ant Colony Optimization. The controllable parameters are the pheromone factor α, the heuristic factor β, and the evaporation rate ρ, with ranges $\alpha, \beta \in [1, 4]$ and $\rho \in [0.1, 0.9]$. We use the direction map

$$d_\alpha = -1, \qquad d_\beta = -1, \qquad d_\rho = +1,$$

reflecting that stronger pheromone and heuristic weighting reinforce known good paths (exploitation), while a higher evaporation rate weakens historical bias and promotes the discovery of new paths (exploration) (Dorigo et al., 2006). Diversity features in Table 1 are adapted to the path representation: the unique solution ratio counts distinct constructed paths, and the mean pairwise distance uses the edge-based symmetric difference between paths.

Modularity. The same scoring and coordination interfaces are retained across the three evaluated metaheuristic families, demonstrating architectural modularity rather than universal solver independence. The feature extractor, representation-specific distance metric, parameter domains, and direction priors remain solver-dependent and must reflect the target search procedure.

F PROMPT TEMPLATES

F.1 NOTES ON PROMPT DESIGN

Symmetric structure. The two prompts are deliberately symmetric in length, formatting, and rubric granularity. This reduces the risk that one assessor systematically reports more urgent scores than the other due to prompt-induced bias rather than the underlying search state.

Stage-relative urgency. Both rubrics instruct the assessor to judge urgency *relative to the current search stage*. Without this instruction, both assessors tend to anchor on absolute thresholds (e.g., always reporting high exploration urgency when diversity is below a fixed level), which produces less informative scores in late stages of a run when one direction has already become dominant.

Output schema and parsing. The single-line JSON output is parsed by $\mathcal{D}_{\text{score}}$. If the response cannot be parsed into a valid integer in $\{0, \ldots, 10\}$, the score is deterministically set to the neutral value 5; we observed parsing failures in less than 1% of decision points across all experiments. The `rationale` field is logged for analysis but is not used by the coordinator.

No few-shot examples. We deliberately do not include few-shot examples of past decisions in the prompt. The decision memory already provides recent decision history in tabular form, and additional hand-crafted examples risked biasing the assessors toward those examples' specific score distributions. We confirmed in preliminary experiments that adding few-shot examples did not improve performance and occasionally degraded it.

This appendix provides the prompt templates used for the two LLM assessors. Both prompts share the same overall structure (role, feature block, temporal block, decision memory, output schema) and differ only in (i) the role description, (ii) the feature group provided, and (iii) the rubric specialization. Variables in angle brackets (e.g., `<feature_value>`) are substituted at runtime by the controller. Feature values follow the feature-specific scaling described in Section 4.2: quantities bounded by definition retain their analytical scales, standardized statistics use their conventional units, and counts or scale-free ratios remain on their native scales. No common fitted min–max normalization is applied.

F.2 Explorer Assessor Prompt

Explorer Assessor

[Role]
You are an Explorer assessor for a population-based metaheuristic algorithm (<algorithm_name>) solving an instance of <problem_type>. Your task is to rate how urgently the search currently requires *additional exploration*, defined as diversifying the population to discover new regions of the search space.

[Diversity Features]
The current population shows the following diversity-related signals:

- Unique solution ratio: <u_value> (bounded in $[0, 1]$ by definition)
- Mean pairwise distance: <d_value> (task-specific distance scale)
- Elite dispersion ratio: <D_elite_value> (scale-free ratio)

<validity_flags_if_any>

[Temporal Context]
The search is currently at:

- Generation ratio: <rho_t> (i.e., <t>/<T> of budget consumed)
- Stagnation count: <sigma> consecutive generations without best-so-far improvement
- Recent improvement rate: <delta_best> over the trailing window of <w> generations
- Mean-fitness trend: <V> (>1 indicates recent population-level improvement)

[Recent History]
The most recent <L> decision points are summarized below: t=<t_i>
lambda: <lambda_old> → <lambda_new>
outcome: delta_best=<...>, delta_D=<...>
scores: u_E=<...>, u_X=<...>
...
[Rubric]
Rate the exploration urgency on the integer scale 0-10:

- 0–2: low urgency (population is sufficiently diverse, no clear signs of stagnation)
- 3–4: mild urgency (slight diversity loss or minor stagnation)
- 5–6: moderate urgency (noticeable diversity loss or sustained stagnation)
- 7–8: high urgency (clear diversity collapse, prolonged stagnation, or both)
- 9–10: critical urgency (severe diversity collapse, long stagnation, or strong evidence of premature convergence)

Judge urgency *relative to the current search stage*: low diversity early in the run typically warrants higher urgency than the same level near the end of the run, especially when exploitation is producing improvements.

[Output Format]
Respond with exactly one JSON object on a single line, with no additional text before or after: {"score": <integer 0–10>, "rationale": "<one short sentence>"} The rationale field is for logging only and is not used by the controller.

F.3 Exploiter Assessor Prompt

Exploiter Assessor

[Role]
You are an Exploiter assessor for a population-based metaheuristic algorithm (<algorithm_name>) solving an instance of <problem_type>. Your task is to rate how urgently the search currently requires *additional exploitation*, defined as intensifying refinement around already-promising regions of the search space.

[Fitness-Quality Features]
The current population shows the following fitness-quality signals:

- Fitness–distance correlation: <FDC_value> (absolute Pearson correlation in $[0, 1]$)
- Elite gap: <g_value> (standardized gap)
- Fitness skewness: <S_value> (standardized skewness)

<validity_flags_if_any>
[Temporal Context]
The search is currently at:

- Generation ratio: <rho_t> (i.e., <t>/<T> of budget consumed)
- Stagnation count: <sigma> consecutive generations without best-so-far improvement
- Recent improvement rate: <delta_best> over the trailing window of <w> generations
- Mean-fitness trend: <V> (>1 indicates recent population-level improvement)

[Recent History]
The most recent <L> decision points are summarized below: t=<t_i>
lambda: <lambda_old> → <lambda_new>
outcome: delta_best=<...>, delta_D=<...>
scores: u_E=<...>, u_X=<...>
...
[Rubric]
Rate the exploitation urgency on the integer scale 0–10:

- 0–2: low urgency (no exploitable structure; landscape appears rugged or random)

- 3–4: mild urgency (weak structure, refinement may not yet be productive)
- 5–6: moderate urgency (some structure visible; refinement is reasonable)
- 7–8: high urgency (clear funnel-like structure or strong elite gap; refinement is likely productive)
- 9–10: critical urgency (highly exploitable landscape; failure to refine wastes the current opportunity)

Judge urgency *relative to the current search stage*: strong exploitable structure late in the run typically warrants higher urgency than the same structure early on, when exploration may still pay off.

[Output Format]
Respond with exactly one JSON object on a single line, with no additional text before or after: {"score": <integer 0–10>, "rationale": "<one short sentence>"} The rationale field is for logging only and is not used by the controller.

G AI USE STATEMENT

During the preparation of this work, we used generative AI tools, including ChatGPT and Claude, to assist with language refinement, manuscript organization, literature and reference checking, and limited feedback on presentation and result interpretation. All final methodological decisions, experimental design, reported results, and conclusions were determined and verified by the authors. Separately, LLMs are components of the proposed HypaEvo method and are used directly in the reported experiments.