EKO: EVOLVING KNOWLEDGE FOR INTERPRETABLE EXPENSIVE MULTIOBJECTIVE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Evolutionary algorithms and surrogate-based methods lack interpretability because they treat expensive multiobjective optimization problems as a black-box numerical search. However, real world problems have valuable but unused domain background that can guide search under scarce evaluations, including variable meanings and known mechanisms. Although large language model (LLM) optimizer can use domain background, existing methods neither inherit nor validate their reasoning, so fail to accumulate correct domain knowledge. We propose EKO, an evolutionary framework whose genetic representation is knowledge, which records reasoning and its validation outcomes in natural language. EKO implements knowledge evolution through three agents: the Selector chooses parent knowledge, the Generator produces offspring and new knowledge, and the Reflector validates this knowledge. EKO repeatedly updates reasoning to build global knowledge, delivering beyond a Pareto set an interpretable characterization of the problem landscape with supporting evidence. Since knowledge evolution is independent of variable representation, changing only the problem description in natural language avoids designing specialized operators for different variable types. Experiments on benchmarks from different domains demonstrate that EKO achieves competitive performance and adaptability while remaining interpretable. Ablation studies and knowledge audits confirm component effectiveness and knowledge traceability.

1 INTRODUCTION

Expensive multiobjective optimization problems (EMOPs) appear in real-world engineering design: vehicle crashworthiness simulation (Liao et al., 2008), biological fermentation yield optimization (Birol et al., 2002), and materials discovery (Gopakumar et al., 2018) each require hours to days per function evaluation, while decision makers need Pareto-optimal solutions under tight evaluation budgets (Liang et al., 2024). Formally, the goal is to approximate the Pareto front over feasible space $\mathcal{X}$ (continuous, combinatorial, or code-level) under a small evaluation budget T (e.g., $T = 50$):

$$\min_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})), \tag{1}$$

These problems are not merely numerical: their objective functions carry physical semantics (variable-objective mechanism couplings, engineering constraint meanings, and reaction kinetics) that domain experts use when reasoning about candidate designs. EKO is designed for settings with costly evaluations: it uses LLM inference to reduce the number of real evaluations, assuming that simulation, laboratory, or human assessment cost dominates the added LLM cost. Under a small budget, a useful optimizer should exploit such information, but existing numerical optimization methods do not use it in their search.

Evolutionary algorithms (EAs) such as NSGA-II (Deb et al., 2002), MOEA/D (Zhang & Li, 2007), and SMS-EMOA (Beume et al., 2007) use numerical operators such as simulated binary crossover (SBX) and polynomial mutation, treating the problem as a black box and producing no reusable understanding. Bayesian optimization (BO) methods such as qEHVI (Daulton et al., 2020) and qLogEHVI (Ament et al., 2023) fit statistical surrogates to observations, but their acquisition functions do not directly use problem physics. LLM-based optimizers (Liu et al., 2024; Schwanke et al., 2026; Yuan et al., 2026; Chen et al., 2026b;a; Lu et al., 2026; Wang et al., 2025; Zhou et al., 2026;

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.

Abhyankar et al., 2026) use language models to generate candidates or act as operators, but their reasoning is typically one-shot: it is neither inherited across generations nor validated against real evaluations. Thus, existing methods do not turn domain-informed reasoning into knowledge that can be accumulated and corrected during optimization.

Our insight is that LLM reasoning (Wei et al., 2022) can be captured as inheritable knowledge. We define knowledge as a natural-language record of the reasoning behind a solution and its validation outcomes. As these validated records accumulate, EKO builds an interpretable characterization of the problem landscape while passing knowledge to later generations. Because this knowledge evolution protocol is independent of variable representation, EKO can apply to continuous, combinatorial, and other optimization problems without designing specialized operators.

EKO implements knowledge evolution through three specialized agents. The **Selector** chooses parent knowledge and search directive from the evaluated parent pool and evolution state. The **Generator** retrieves domain background, parent knowledge, and global knowledge to generate offspring with unvalidated reasoning records. The **Reflector** evaluates offspring, validates the registered hypothesis against relative objective changes from parents, and updates global knowledge and the evolution state for later search. This cycle lets EKO use domain semantics to guide generation, correct knowledge with evaluation outcomes, and maintain an interpretable knowledge archive.

Our contributions are as follows:

1. We define knowledge in evolution as the inheritable natural-language record of reasoning and validation outcomes. This representation preserves LLM reasoning across generations, enabling optimization to accumulate problem understanding and provide an interpretable characterization of the problem landscape.
2. We use three LLM agents—Selector, Generator, and Reflector—to replace parent selection, offspring generation, and evaluation feedback in the evolutionary process. Their separated responsibilities form a controlled, evidence-calibrated loop that ensures validated reasoning becomes inheritable knowledge.
3. We evaluate EKO on six continuous real-world tasks for the main performance comparison and on binary, categorical, and permutation tasks without representation-specific numerical operators. Knowledge validation audits and ablations assess the reliability of the generated knowledge and the role of each knowledge source.

2 Related Work

Traditional Expensive Multi-Objective Optimization. MOEAs including NSGA-II (Deb et al., 2002), NSGA-III (Deb & Jain, 2014), MOEA/D (Zhang & Li, 2007), and SMS-EMOA (Beume et al., 2007) evolve populations via numerical operators (SBX, polynomial mutation, decomposition-based selection) but require thousands of evaluations. Surrogate-assisted EAs (SAEAs) fit GP/RBF/SVM surrogates to reduce evaluations (Liang et al., 2024) but require surrogate retraining and produce no interpretable knowledge. BO methods such as EHVI (Emmerich et al., 2006), qEHVI (Daulton et al., 2020), qLogEHVI (Ament et al., 2023), and ParEGO (Knowles, 2006) use Gaussian process surrogates with acquisition functions, but their numerical surrogates do not encode problem physics. Recent work does bring physics into Bayesian optimization, for example by combining it with physical domain knowledge (Christen & Kirchartz, 2026), but such knowledge enters as numerical model structure, kernels, or constraints rather than as natural-language reasoning. In all these methods, feedback remains numerical populations, surrogate parameters, or acquisition values rather than explicit and inspectable statements about the mechanisms and objective trade-offs of real-world systems (Jones et al., 1998; Nezami & Anahideh, 2024).

LLM-Based Optimization. Recent work uses LLMs in two directions. The first uses LLMs as candidate generators, surrogates, or priors: LLAMBO (Liu et al., 2024) elicits candidates and surrogate priors from natural language descriptions of the optimization task. HOLLM (Schwanke et al., 2026) couples LLM guidance with hierarchical search-space partitioning to focus search on promising subregions. LGBO (Yuan et al., 2026) translates LLM-derived preferences into a guidance signal that is injected into the BO loop. LABO (Chen et al., 2026b) uses LLM guidance to coordinate broad, low-cost exploration with selective high fidelity experimentation. Evidence-

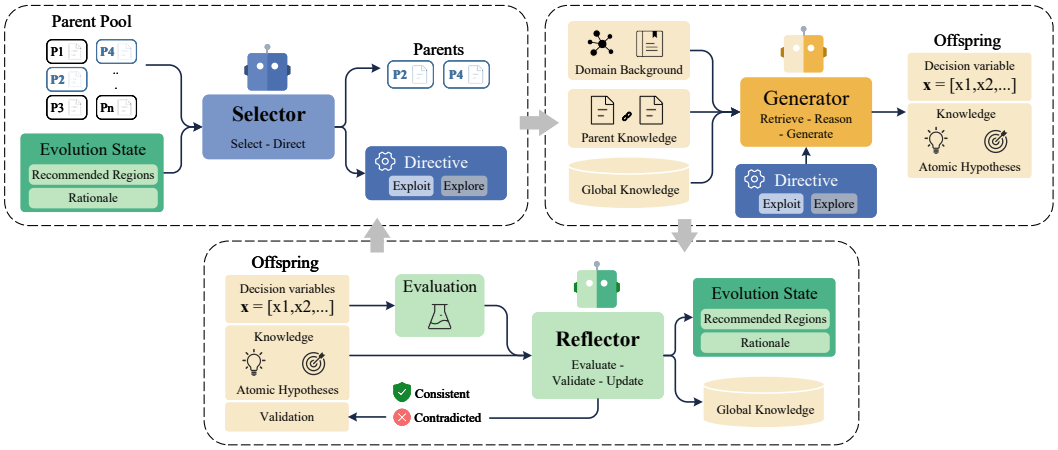

Figure 1: EKO's knowledge evolution loop with three agents. The Selector reads the parent pool and evolution state, then chooses parent knowledge and issues an exploit or explore directive. The Generator retrieves domain background, parent knowledge, and global knowledge to generate an offspring with an unvalidated reasoning record. After evaluation and validation, the Reflector completes the record, then updates the evolution state and global knowledge for the next round.

Gated (Chen et al., 2026a) calibrates LLM-generated priors against empirical evidence before they influence multiobjective BO. R2SAEA (Lu et al., 2026) finetunes LLMs as relational surrogates to model relations among decision variables and objectives. These methods often leave reasoning in one-shot or ever-growing prompt contexts, neither consolidating it into reusable knowledge nor closing the loop to calibrate it continually with real evaluations. The second direction replaces EA operators with LLMs: MOLLEO (Wang et al., 2025) uses LLMs to propose chemically meaningful variation operators for evolutionary search over chemical space. ToolMol (Zhou et al., 2026) equips an LLM-driven genetic algorithm with tool calls to generate and evaluate drug candidates under multiple objectives. LLEMA (Abhyankar et al., 2026) combines LLM-guided evolutionary search with a memory mechanism for multi-objective materials discovery. These methods inherit decision variables but not the reasoning behind them. Their memory pools store no validated expectations.

Optimization Based on Domain Knowledge. Process knowledge has been used to adapt evolutionary search strategies (Zuo et al., 2024). Physics-informed evolutionary transfer optimization further encodes material validity, properties, and energy as physical objectives to guide multiobjective crystal design (Chen et al., 2025). In scientific discovery, AI Hilbert combines experimental data with formal background theory to derive mathematically certified scientific models (Cory-Wright et al., 2024), whereas Co-Scientist iteratively generates, critiques, and refines hypotheses against prior scientific evidence for experimental validation (Gottweis et al., 2026). These approaches encode guidance as strategy models, physical objectives, constraints, or formal theories, rather than as an evolving archive of natural-language hypotheses tied to individual search transitions. EKO instead extracts LLM reasoning as inheritable natural-language knowledge, validates it with real evaluations, and accumulates it across generations to evolve problem understanding alongside solutions.

3 EKO: Knowledge Evolution with LLMs

Overview: Three-Agent Architecture. EKO delegates knowledge evolution to three agents with separated responsibilities, replacing three stages of a conventional evolutionary loop. The Selector chooses parents and issues a search directive. The Generator randomly anchors one selected parent and generates an offspring from retrieved knowledge of parents and reasoning. The Reflector validates the hypotheses registered in reasoning, then updates global knowledge and the evolution state. The validated and updated knowledge will then enter the next round of inheritance. EKO makes the optimization process interpretable by making transparent the evidence chain from retrieved knowledge and reasoning to the generated solution, its validation outcome, and subsequent knowledge.

Algorithm 1 EKO Main Loop

Input: problem interface $\mathcal{I}$, domain background B, evaluation budget T
Output: nondominated set S^*, global knowledge archive $\mathcal{A}_t$

1: Initialize: solution pool S_0, evolution state z_0, global knowledge archive $\mathcal{A}_0$
2: $f(S_0) \leftarrow \textsc{Evaluate}(S_0, \mathcal{I})$; $t \leftarrow |S_0|$
3: **while** $t < T$ **do**
4: $\quad (P_t, q_t) \leftarrow \textsc{Selector}(S_t, z_t)$
5: $\quad (o_t, \ell_t^-) \leftarrow \textsc{Generator}(\mathcal{I}, B, P_t, \mathcal{A}_t, q_t)$
6: $\quad (o_t, \ell_t^-) \leftarrow \textsc{CheckFeasibility}((o_t, \ell_t^-), \mathcal{I})$
7: $\quad$ **while** o_t is infeasible **do**
8: $\quad\quad (o_t, \ell_t^-) \leftarrow \textsc{Generator}(\mathcal{I}, B, P_t, \mathcal{A}_t, q_t)$
9: $\quad\quad (o_t, \ell_t^-) \leftarrow \textsc{CheckFeasibility}((o_t, \ell_t^-), \mathcal{I})$
10: $\quad$ **end while**
11: $\quad f(o_t) \leftarrow \textsc{Evaluate}(o_t, \mathcal{I})$
12: $\quad \ell_t \leftarrow \textsc{Reflector.Validate}(\ell_t^-, f(o_t), f(a_t))$
13: $\quad (\mathcal{A}_{t+1}, z_{t+1}) \leftarrow \textsc{Reflector.Update}(\mathcal{A}_t, z_t, \ell_t)$
14: $\quad S_{t+1} \leftarrow S_t \cup \{o_t\}$; $t \leftarrow t + 1$
15: **end while**
16: **return** $\textsc{Nondominated}(S_t), \mathcal{A}_t$

For each offspring, the Reflector aggregates these human-readable validated reasoning into global knowledge, so that every archived conclusion can be traced back to its prediction and objective evidence. Figure 1 illustrates the resulting three-agent knowledge evolution architecture.

EKO uses the same LLM $\mathcal{M}$ for all three agents. Their distinct responsibilities are specified by prompt templates Γ_{sel}, Γ_{gen}, and Γ_{ref}. Each template specifies an agent's role, accessible inputs, required outputs, and output format. The complete templates and schemas are provided in the supplementary material.

Algorithm 1 formalizes the round-level control flow illustrated of EKO. It takes a problem interface $\mathcal{I}$, domain background B, and an evaluation budget T as input. The fixed domain background B is compiled before optimization from publicly available problem descriptions and qualitative domain regularities. Its contents are provided in the supplementary material. The problem interface $\mathcal{I}$ specifies the variable representation, feasibility checker, and objective evaluation procedure for a problem. The LLM Generator serves as the offspring operator by proposing edits through this interface, rather than using type-specific crossover or mutation. The feasibility checker validates representations and rejects invalid ones. Thus, changing $\mathcal{I}$ adapts EKO to problems with different variable representations without designing operators for each variable type. Beyond a nondominated set S^*, EKO returns a global knowledge archive $\mathcal{A}_t$ that provides an interpretable characterization of the problem landscape.

Selector. The Selector call at Algorithm 1, line 4, reads the evaluated parent pool S_t and evolution state z_t. z_t records recommended regions, saturated regions, and short rationale from the preceding round. Each member of S_t contains decision variables, objective values, and validated solution-level knowledge. The Selector uses the regions recorded in z_t to choose two parents from S_t, and returns a search directive q_t. It is implemented by LLM $\mathcal{M}$ under the following prompt contract:

$$(P_t, q_t) = \mathcal{M}(S_t, z_t, \Gamma_{\text{sel}}), \tag{2}$$

where $|P_t| = 2$ and $q_t \in \{\textit{exploit}, \textit{explore}\}$ is a predefined natural-language prompt that guides offspring generation. The *exploit* prompt directs the Generator to refine promising local regions with small, targeted edits, whereas the *explore* prompt directs it to make broader edits toward underexplored regions or trade-offs. The two selected parents provide the candidate solutions and knowledge from which the subsequent offspring is derived. Γ_{sel} defines the Selector's role, how it uses the parent pool ,evolution state, and the rules for returning two selected parents and one of the two directives.

Generator and unvalidated reasoning. The Generator call at Algorithm 1, line 5, receives problem interface $\mathcal{I}$, domain background B, selected parents P_t, global knowledge archive $\mathcal{A}_t$, and search directive q_t. Let

$$I_t = (B, P_t, \mathcal{A}_t) \tag{3}$$

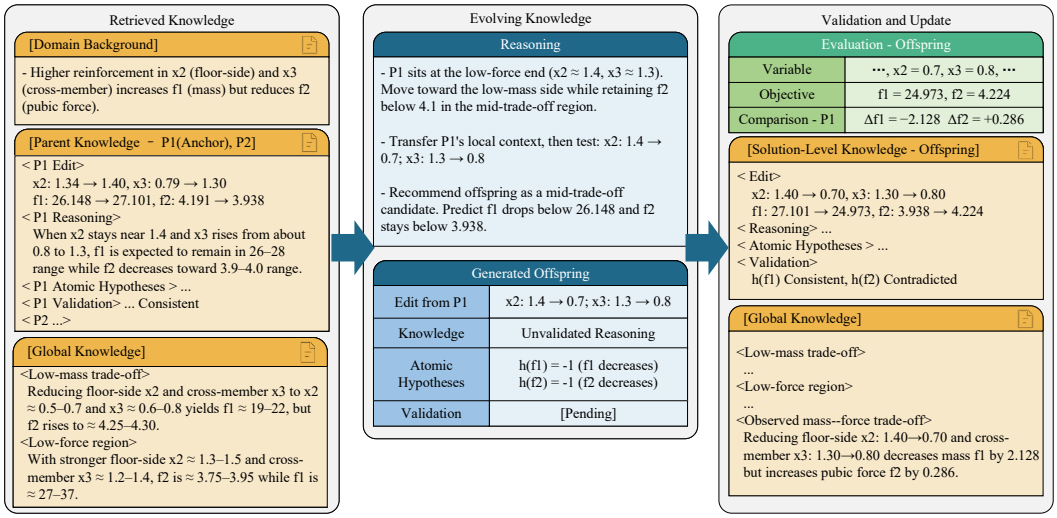

Figure 2: A simplified single-round example of knowledge evolution in CarSideImpact. Left: retrieved domain background, knowledge from anchor parent P1(anchor) and P2, and two global knowledge statements. Center: reasoning, the edit from P1, and the generated offspring with pending atomic-hypothesis validation. Right: evaluate the offspring and then calculate the change relative to P1, completed solution-level knowledge with consistent and contradicted hypotheses, and global knowledge updated by adding an observed mass–force trade-off. For clarity, the figure restricts reasoning to x_2 and x_3.

collect the retrieved knowledge. The Generator is implemented as

$$(o_t, \ell_t^-) = \mathcal{M}(\mathcal{I}, I_t, q_t, \Gamma_{\text{gen}}), \tag{4}$$

where o_t is the offspring generated in round t and ℓ_t^- is its unvalidated reasoning record. The record is

$$\ell_t^- = (a_t, e_t, r_t, \mathcal{H}_t), \tag{5}$$

where the anchor parent $a_t \in P_t$ is selected uniformly at random. It serves as the starting point for the Generator's reasoning and offspring construction, and provides the reference for validating the registered hypotheses. The edit e_t specifies the changed decision-variable values from a_t to o_t, while r_t records the supporting reasoning. The Generator derives atomic hypotheses from the predicted objective effects stated in its reasoning r_t for edit e_t, then registers them according to the output schema specified by Γ_{gen}. Let $J_t \subseteq \{1, \ldots, m\}$ be the set of objectives for which r_t predicts that the edit has an effect. For each $j \in J_t$, it registers one directional atomic hypothesis

$$h_{t,j} \in \{-1, +1\}, \qquad \mathcal{H}_t = \{h_{t,j} \mid j \in J_t\}, \tag{6}$$

where $h_{t,j} = -1$ predicts a decrease in objective j, and $h_{t,j} = +1$ predicts an increase. They are later compared with the observed relative change $\Delta_{t,j} = f_j(o_t) - f_j(a_t)$. Thus, the Generator registers hypotheses before evaluation in a fixed, machine-readable form that supports deterministic validation. Γ_{gen} defines the Generator's role, how it uses $\mathcal{I}$, I_t, and q_t, and the rules for returning one offspring and its unvalidated reasoning record. Guided by Γ_{gen}, the Generator follows a *Retrieve–Reason–Generate* procedure: it reasons over the retrieved knowledge, generates one offspring, and registers its atomic hypotheses before real evaluation. The deterministic CHECKFEASIBILITY interface checks the generated offspring before real evaluation (Algorithm 1, line 6). If the proposal is infeasible, the Generator retries with the same round inputs for at most three attempts. Failed retries consume LLM generation tokens but not objective evaluations, and the round is skipped if all attempts fail.

Reflector, validation, and knowledge update. After the algorithm evaluates the feasible offspring at Algorithm 1, line 11, the Reflector validates its hypotheses and updates knowledge at lines 12–13. For the feasible offspring o_t, the anchor parent is a_t. After evaluation, the change from the anchor parent for every objective j is

$$\Delta_{t,j} = f_j(o_t) - f_j(a_t). \tag{7}$$

For each atomic hypothesis $h_{t,j} \in \mathcal{H}_t$, its binary validation outcome $v_{t,j}$ is computed:

$$v_{t,j} = \begin{cases} \texttt{consistent}, & h_{t,j}\Delta_{t,j} > 0, \\ \texttt{contradicted}, & h_{t,j}\Delta_{t,j} \leq 0. \end{cases} \tag{8}$$

The validated solution-level knowledge is

$$\ell_t = (\ell_t^-, \mathcal{V}_t), \qquad \mathcal{V}_t = \{v_{t,j} \mid j \in J_t\}. \tag{9}$$

By retaining every original atomic prediction and its deterministic validation outcome, ℓ_t distinguishes predictions supported by evaluation from those contradicted by it. Only the validated solution-level knowledge ℓ_t can update global knowledge and the evolution state. The update at Algorithm 1, line 13, is implemented by

$$(\mathcal{A}_{t+1}, z_{t+1}) = \mathcal{M}(\mathcal{A}_t, \ell_t, z_t, \Gamma_{\text{ref}}). \tag{10}$$

Γ_{ref} defines the Reflector's role, how it uses $\mathcal{A}_t$, ℓ_t, z_t, the rules and format for explaining and aggregating completed solution-level knowledge into global knowledge and the evolution state. The evolution state z_t is a structured natural-language record containing recommended regions, saturated regions, and a short rationale. Consequently, the global knowledge archive outputs reusable, human-readable statements about objective trade-offs and variable operating regions, each grounded in completed records rather than raw LLM assertions. The Reflector is required not to add unsupported statements to the archive, and every update must be based on existing validated reasoning. Global knowledge is supplied to subsequent Generator calls, whereas the evolution state is supplied to the next Selector call.

Illustrative knowledge evolution event. Figure 2 presents a simplified example of knowledge evolution in CarSideImpact, omitting the directive, evolution state, and unrelated variables for clarity. P1 is the randomly selected anchor parent and P2 supplies additional contextual knowledge. The event makes EKO's interpretable outputs concrete: the unvalidated record explains the edit from P1 and its predicted effects, while the completed record and archive update preserve their evaluation evidence. It instantiates the retrieved knowledge $I_t = (B, P_t, \mathcal{A}_t)$. B states that increasing the floor-side thickness x_2 and cross-member thickness x_3 raises mass F_1 but reduces pubic force F_2. The selected-parent knowledge supplies P1's local observation around $x_2 \approx 1.4$ and $x_3 \approx 1.3$, together with P2 as contextual evidence. The archive contains two inherited natural-language statements with numerical ranges describing low-mass and low-force trade-offs. Guided by the search directive, the Generator transfers P1's local context to propose an offspring with $x_2 : 1.4 \to 0.7$ and $x_3 : 1.3 \to 0.8$. Its unvalidated reasoning record registers two atomic hypotheses: the edit is expected to decrease F_1 and F_2 relative to P1.

The algorithm evaluates the offspring as $(F_1, F_2) = (24.973, 4.224)$ and computes $\Delta_{t,1} = -2.128$ and $\Delta_{t,2} = +0.286$ relative to P1. The F_1 hypothesis is *consistent*, whereas the F_2 hypothesis is *contradicted*, because the edit raises F_2 rather than producing the registered decrease. The completed solution-level knowledge preserves the observed transition from P1, both original atomic predictions, and their respective evaluation outcomes. The global knowledge update retains the two inherited statements and adds a mass–force trade-off statement with the observed variable changes and objective differences. Thus, every archive statement remains interpretable through its supporting completed records. The supplementary material provides an original execution log of a complete EKO iteration, including the preserved agent inputs and outputs from parent selection through validation and knowledge update. This record directly exposes the form of EKO's natural-language knowledge and the evidence trail supporting its interpretability.

4 Experiments

4.1 Experimental Setup

Benchmarks. We use six continuous real-world multiobjective problems as the main performance comparison. CarSideImpact (7 variables / 2 objectives, vehicle crashworthiness) and VehicleSafety (5/3, vehicle safety) are drawn from the BoTorch test-function suite (Balandat et al., 2020). CrossBarrel_MO (4/2, cross-barrel material parameters), Fullerene_MO (3/2, fullerene synthesis parameters), P3HT_MO (5/2, P3HT/CNT formulation ratios), and PCE10_MO (4/2, photovoltaic material

Table 1: Mean final HV (↑) on six continuous multiobjective tasks under $T = 50$ real evaluations. Values are mean ± 95% confidence intervals over 10 seeds. M denotes million, and the best values are bold. Significance of EKO against the best baseline is assessed with a two-sided Wilcoxon signed-rank test on paired per-seed results (* $p < 0.05$, ** $p < 0.01$).

Type	Method	CarSideImpact	VehicleSafety	CrossBarrel_MO	Fullerene_MO	P3HT_MO	PCE10_MO
EA	NSGA-II	13.420 ± 1.016	50.701 ± 4.496	841.65 ± 52.70	2375.65 ± 34.44	5.078 ± 0.673M	260.558 ± 13.37
	MOEA/D	12.488 ± 1.082	46.928 ± 4.162	814.19 ± 68.72	2357.62 ± 46.48	4.524 ± 0.768M	263.175 ± 13.74
BO	EHVI	19.068 ± 0.065	79.122 ± 0.437	1149.45 ± 25.85	2477.64 ± 10.72	7.371 ± 0.358M	312.089 ± 2.343
	qLogEHVI	19.029 ± 0.060	78.724 ± 0.395	1140.05 ± 31.55	2474.49 ± 13.80	7.887 ± 0.183M	311.995 ± 1.961
LLM-based	HOLLM	14.167 ± 0.614	70.117 ± 5.297	949.08 ± 84.89	2374.59 ± 21.04	6.026 ± 0.702M	302.519 ± 17.09
	LLAMBO	14.158 ± 0.475	71.727 ± 5.212	945.13 ± 102.50	2372.16 ± 21.83	5.692 ± 0.765M	318.741 ± 9.66
Ours	**EKO**	**20.285**** ± **0.074**	**85.326**** ± **0.700**	**1153.95** ± **48.94**	**2494.57*** ± **9.68**	**8.210*** ± **0.263M**	**329.789** ± **0.365**

composition ratios) are constructed from the corresponding Olympus scientific datasets (Häse et al., 2021). The domain background provided to EKO contains variable semantics, types, ranges, constraints, objective meanings, and public qualitative domain knowledge.

To assess performance beyond continuous variables, we additionally use MOKP25 (25/2, binary knapsack) (Zitzler & Thiele, 1999), WDBC (30/2, binary feature selection) (Street et al., 1993), LNP3 (5/3, categorical lipid-nanoparticle formulation) (Häse et al., 2021), NB201 (6/2, categorical neural architecture search) (Dong & Yang, 2020), TSP (20/2, permutation bi-objective routing), and TSP52 (52/2, permutation bi-objective routing) (Reinelt, 1991).

Algorithms. **EA**: NSGA-II (Deb et al., 2002) and MOEA/D (Zhang & Li, 2007); **BO**: EHVI (Emmerich et al., 2006) and qLogEHVI (Ament et al., 2023); **LLM-based**: HOLLM (Schwanke et al., 2026) and LLAMBO (Liu et al., 2024). HOLLM combines LLM candidate generation with adaptive search space partitioning, LLAMBO uses LLMs to enhance BO surrogate modeling and candidate sampling. EKO, HOLLM, and LLAMBO receive the same fixed domain background, comprising publicly available problem descriptions and qualitative domain regularities, through their textual context interfaces. The available HOLLM and LLAMBO implementations do not support binary, categorical, or permutation spaces, so they are excluded from the non-continuous comparison. NSGA-II, MOEA/D, EHVI, and qLogEHVI retain their standard numerical black-box interfaces.

Unified protocol. We evaluate all methods in the real world expensive optimization regime. Each method receives a budget of $T = 50$ objective evaluations. We run every method with 10 independent seeds, and for each seed, all methods share the same five-point initial design and other seed-controlled randomness. We use DeepSeek v4-flash as the primary LLM, all LLM-based baselines use the same backend, temperature, token limit, and real-evaluation budget. We report fixed-reference HV (Zitzler & Thiele, 1999), with a common reference point for all methods on each task. The task-specific reference points are fixed before experiments, shared across methods and seeds, and detailed in the supplementary material.

4.2 Main Results

Table 1 reports final HV on the six continuous multiobjective tasks under the common scarce evaluation budget for expensive optimization. EKO attains the highest mean HV on all six tasks, exceeding the strongest non-EKO result by 6.4% on CarSideImpact, 7.8% on VehicleSafety, 0.4% on CrossBarrel_MO, 0.7% on Fullerene_MO, 4.1% on P3HT_MO, and 5.7% on PCE10_MO. Figure 3 shows that EKO remains competitive during early search and continues improving after the BO and LLM-based baselines plateau, yielding the largest late-budget gains on CarSideImpact, VehicleSafety, P3HT_MO, and PCE10_MO. On CrossBarrel_MO and Fullerene_MO, the methods converge to similar final HV, while EKO retains the highest mean HV.

4.3 Cross-Representation Performance

We evaluate EKO on the non-continuous tasks described above to assess performance across binary, categorical, and permutation representations. For each representation, we change only the prob-

Table 2: Mean final HV (↑) on six non-continuous tasks under $T = 50$ real evaluations. Values are mean ± 95% confidence intervals over 10 seeds. Best values are bold, and M and B denote million and billion, respectively. Significance of EKO against the best baseline is assessed with a two-sided Wilcoxon signed-rank test on paired per-seed results (* $p < 0.05$, ** $p < 0.01$).

Type	Task	NSGA-II	qLogEHVI	EKO
Binary	MOKP25	0.933 ± 0.032M	1.225 ± 0.009M	**1.362 ± 0.011M****
	WDBC	21.188 ± 0.960	24.586 ± 0.500	**28.345 ± 0.119****
Categorical	LNP3	0.2169 ± 0.0629	**0.3908 ± 0.0093**	0.3703 ± 0.0077
	NB201	1244.40 ± 71.51	**1346.43 ± 11.07**	1327.55 ± 12.28
Permutation	TSP	1.026 ± 0.038B	1.003 ± 0.032B	**1.137 ± 0.015B****
	TSP52	7.972 ± 0.135B	7.767 ± 0.108B	**8.653 ± 0.067B****

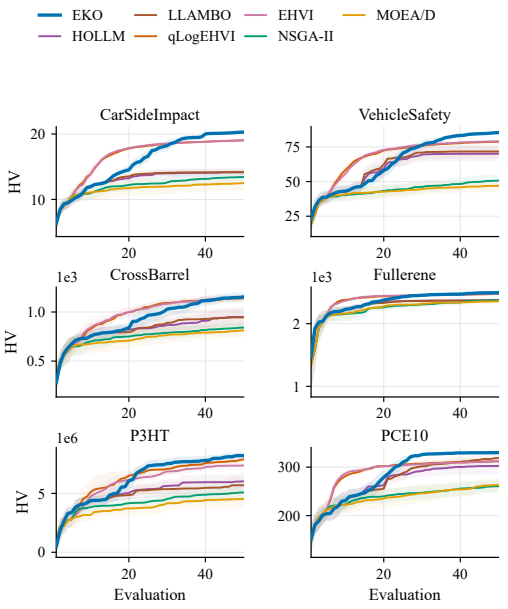

Figure 3: HV convergence curves on the six continuous multiobjective tasks in Table 1 under $T = 50$ real evaluations. Lines show means and shaded regions show 95% confidence intervals over 10 seeds.

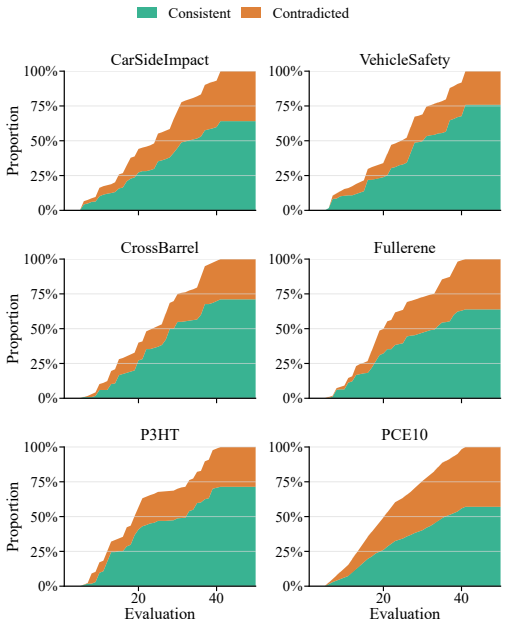

Figure 4: Knowledge audit through aggregated validation outcomes across six tasks. Panels show cumulative proportions over 50 evaluations, normalized by the final number of predictions in each task.

lem interface and domain background. The definition of knowledge and three-agent loop remain unchanged, and EKO introduces no representation-specific numerical operators.

Table 2 reports the final HV obtained over 10 seeds. EKO achieves the best mean HV on four of the six tasks: the two binary tasks MOKP25 and WDBC, and both permutation tasks TSP and TSP52. Its advantage over qLogEHVI is 11.2% on MOKP25, 15.3% on WDBC, 13.3% on TSP, and 11.4% on TSP52, while it exceeds NSGA-II on every task. The categorical tasks are exceptions: qLogEHVI is higher by 1.4% on NB201 and 5.6% on LNP3. These results show that the unchanged knowledge-evolution protocol remains competitive across three non-continuous representations.

4.4 Knowledge Audits

We audit the validation outcomes attached to EKO's registered atomic predictions over the optimization trajectory. Each registered prediction is compared with the subsequent objective evaluation and retained as either consistent or contradicted.

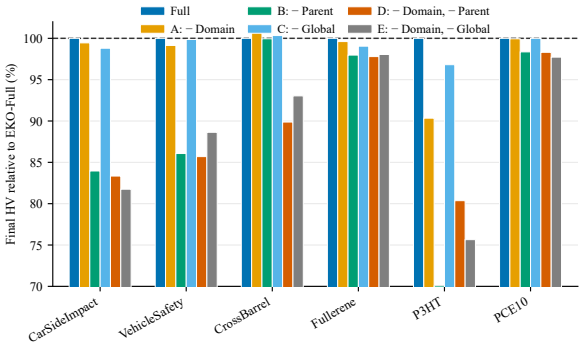

Figure 5: Knowledge source ablation across six tasks. Each bar reports the final HV relative to EKO-Full, averaged over 10 seeds. The dashed line marks the EKO-Full reference (100%).

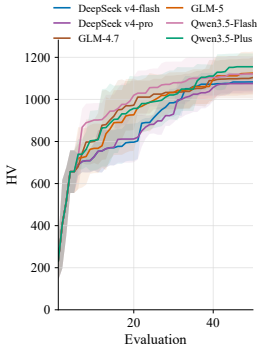

Figure 6: LLM backbone sensitivity on CrossBarrel_MO. Solid curves show the mean HV and shaded bands denote standard deviation.

Figure 4 shows the cumulative proportions of validation outcomes. Consistent outcomes are the largest category for every task. Contradicted outcomes are retained as corrective evidence. These outcomes show that many registered predictions agree with subsequent evaluations, while the retained contradictions provide counterevidence for correcting later knowledge.

4.5 Ablations

The ablation isolates the three knowledge sources in $I_t = (B, P_t, \mathcal{A}_t)$. Figure 5 reports each variant's final HV relative to EKO-Full over 10 seeds. Removing domain background alone barely affects five tasks but lowers HV on P3HT, where task-specific mechanistic guidance matters most. Removing parent knowledge causes the largest losses, reducing performance to 84.0% on CarSideImpact, 86.1% on VehicleSafety, and 70.1% on P3HT_MO, confirming that validated local reasoning is critical under a 50-evaluation budget. Removing global knowledge alone has limited effect, but degrades further when combined with domain background removal. Paired removals degrade more than the corresponding single source removals, showing the three sources are complementary rather than interchangeable.

4.6 LLM Backbone Sensitivity

We evaluate six LLM backbones of EKO on CrossBarrel_MO: DeepSeek v4-flash and v4-pro, GLM-4.7 and GLM-5, and Qwen3.5-Flash and Qwen3.5-Plus. Figure 6 shows the mean HV trajectory for each backbone, with shaded bands denoting one standard deviation. Qwen3.5-Plus achieves the highest final HV (1155.33 ± 51.88). All six backbones show similar early improvement and convergence, indicating that EKO is not highly sensitive to backbone capability.

5 Conclusion

We introduced EKO, an interpretable expensive multiobjective optimization method in which natural-language reasoning is stored as knowledge that later generations can retrieve and revise. The Selector chooses parent knowledge, the Generator proposes candidates together with testable predictions, and the Reflector compares those predictions with the observed objectives. This loop maintains a Pareto set and an evidence-linked knowledge archive without requiring representation-specific numerical operators. Under a shared budget of 50 evaluations, EKO obtains the highest final HV on all six continuous real-world tasks and remains competitive on binary, categorical, and permutation problems after changing only the problem interface and domain background.

The method requires several LLM calls in each optimization round, and its behavior depends on the selected model and prompting configuration. Our evaluation covers small and medium search spaces, leaving large-scale problems unresolved. Future work will study multi-LLM workflows with lower cost, improved validation and retrieval rules.

AI USE STATEMENT

In this work, we used generative AI tools (Claude Code) as assistants during the research and writing process. For tasks subject to required disclosure, we used generative AI to implement parts of the experimental code, including baseline implementations and evaluation scripts, to debug that code, and to translate parts of the manuscript text. Parts of the natural-language domain background used in our experiments were drafted with generative AI assistance from publicly available problem descriptions and were manually reviewed by the authors before use. We did not use generative AI to generate synthetic data sets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims or assist in writing proofs, to design or give feedback on the methodology and experiments, to interpret results, or to clean or reformat datasets; the remaining required-disclosure tasks are not applicable to this work. For tasks under recommended disclosure, we used generative AI to draft and edit parts of the paper, to summarize and cross-check the related literature, to format and verify bibliographic entries, and to write the plotting code used to produce the experimental result figures. The LLM agents inside EKO (the Selector, Generator, and Reflector) are the object of study of this paper and are evaluated in our experiments; this statement concerns our own use of AI tools in producing the submission, which is separate from the LLM backbones used by EKO itself. We have reviewed all AI-assisted work, and we take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our results in the main paper and the supplementary material. The supplementary material specifies the twelve multiobjective tasks, including their variable types, ranges, and objective formulations, together with the data sources and the interpolation procedures used to construct the scientific-materials tasks, and it describes all baselines and their configurations. It further contains the complete prompt templates and output schemas of the Selector, Generator, and Reflector, the domain background supplied to the LLM-based methods, and a full execution trace of the illustrative knowledge evolution event reported in Figure 2. The optimization protocol is described in Section 4.1: all methods receive the same budget of $T = 50$ real objective evaluations, the same five-point initial design per seed, and the same reference point for hypervolume, and all LLM-based methods use the same backbone, temperature, and token limits; results are averaged over ten seeds (100, 200, . . . , 1000). Per-task final-HV statistics with 95% confidence intervals are reported in Tables 1 and 2; the convergence curves for the non-continuous tasks, with 95% confidence bands, are reported in the supplementary material. An anonymized implementation of EKO, including the experimental pipeline and the configuration files used to produce the reported results, is provided as a `.zip` archive in the supplementary material.

REFERENCES

Nikhil Abhyankar, Sanchit Kabra, Saaketh Desai, and Chandan K. Reddy. LLEMA: Evolutionary search with LLMs for multi-objective materials discovery. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization, 2020. URL `https://arxiv.org/abs/1910.06403`.

Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

Gülnur Birol, Cenk Ündey, Satish J. Parulekar, and Ali Çınar. A morphologically structured model for penicillin production. *Biotechnology and Bioengineering*, 77(5):538–552, 2002.

Cheng Chen, Haokai Hong, Wanyu Lin, and Kay Chen Tan. A physics-informed evolutionary transfer optimization framework for material design. In *2025 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–9, 2025. doi: 10.1109/CEC65147.2025.11042916.

Jiangyu Chen, Yi Ban, and Tianfan Fu. CALIPER: Evidence-gated LLM prior layers for multi-objective Bayesian optimization, 2026a.

Zhuo Chen, Xinzhe Yuan, Jianshu Zhang, Jinzong Dong, Ruichen Zhou, Yingchun Niu, Tianhang Zhou, Yu Yang Fredrik Liu, Yuqiang Li, Nanyang Ye, and Qinying Gu. LABO: LLM-accelerated Bayesian optimization through broad exploration and selective experimentation. In *Forty-third International Conference on Machine Learning*, 2026b. URL `https://openreview.net/forum?id=KxLNNxz9T9`.

Leonard Christen and Thomas Kirchartz. Model-based Bayesian optimization for organic photovoltaics: Combining Bayesian optimization with physical domain knowledge. *Advanced Energy Materials*, pp. e71249, 2026. doi: 10.1002/aenm.71249.

Ryan Cory-Wright, Cristina Cornelio, Sanjeeb Dash, Bachir El Khadir, and Lior Horesh. Evolving scientific discovery by unifying data and background knowledge with AI Hilbert. *Nature Communications*, 15(1):5922, 2024. doi: 10.1038/s41467-024-50074-w.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

Michael T. M. Emmerich, Kyriakos C. Giannakoglou, and Boris Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006. doi: 10.1109/TEVC.2005.859463.

Abhijith M. Gopakumar, Prasanna V. Balachandran, Dezhen Xue, James E. Gubernatis, and Turab Lookman. Multi-objective optimization for materials discovery via adaptive design. *Scientific Reports*, 8(1):3738, 2018. doi: 10.1038/s41598-018-21936-3.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, et al. Accelerating scientific discovery with co-scientist. *Nature*, 655(8122):487–496, 2026. doi: 10.1038/s41586-026-10644-y.

Florian Häse, Matteo Aldeghi, Riley J. Hickman, Loïc M. Roch, Melodie Christensen, Elena Liles, Jason E. Hein, and Alán Aspuru-Guzik. Olympus: A benchmarking framework for noisy optimization and experiment planning. *Machine Learning: Science and Technology*, 2(3):035021, 2021.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998. doi: 10.1023/A:1008306431147.

Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1): 50–66, 2006.

Jing Liang, Yahang Lou, Mingyuan Yu, Ying Bi, and Kunjie Yu. A survey of surrogate-assisted evolutionary algorithms for expensive optimization. *Journal of Membrane Computing*, 7(2):108–127, 2024. doi: 10.1007/s41965-024-00165-w.

Xingtao Liao, Qing Li, Xujing Yang, Wei Li, and Weigang Zhang. A two-stage multi-objective optimisation of vehicle crashworthiness under frontal impact. *International Journal of Crashworthiness*, 13(3):279–288, 2008. doi: 10.1080/13588260801933659.

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

Ye Lu, Bingdong Li, Aimin Zhou, and Hao Hao. Relation reasoning with LLMs in expensive optimization, 2026.

Nazanin Nezami and Hadis Anahideh. Building trust in black-box optimization: A comprehensive framework for explainability, 2024. URL https://arxiv.org/abs/2410.14573.

Gerhard Reinelt. TSPLIB—a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Andrej Schwanke, Lyubomir Ivanov, David Salinas, Fabio Ferreira, Aaron Klein, Frank Hutter, and Arber Zela. Improving LLM-based global optimization with search space partitioning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=y6nhcCdQYd.

W. Nick Street, William H. Wolberg, and Olvi L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, volume 1905, pp. 861–870, 1993. doi: 10.1117/12.148698.

Haorui Wang et al. Efficient evolutionary search over chemical space with large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xinzhe Yuan, Zhuo Chen, Jianshu Zhang, Huan Xiong, Nanyang Ye, Yuqiang Li, and Qinying Gu. Unleashing LLMs in Bayesian optimization: Preference-guided framework for scientific discovery. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

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

Andrew Y. Zhou, Sharvaree Vadgama, Sumanth Varambally, Peter Eckmann, Michael K. Gilson, and Rose Yu. ToolMol: Evolutionary agentic framework for multi-objective drug discovery, 2026.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Mingcheng Zuo, Dunwei Gong, Yan Wang, Xianming Ye, Bo Zeng, and Fanlin Meng. Process knowledge-guided autonomous evolutionary optimization for constrained multiobjective problems. *IEEE Transactions on Evolutionary Computation*, 28(1):193–207, 2024. doi: 10.1109/TEVC.2023.3243109.

SUPPLEMENTARY MATERIAL

A DETAILS ON MULTIOBJECTIVE TASKS

We evaluate EKO on twelve multiobjective tasks covering continuous engineering design, scientific-materials optimization, and binary, categorical, and permutation decision spaces. All experiments use a common budget of $T = 50$ real objective evaluations, a frozen five-point initial design for each task–seed pair, and ten seeds $(100, 200, \ldots, 1000)$. All objectives are expressed in minimization form; objectives originally defined for maximization are negated before optimization. EKO operates internally in a $[0, 100]$ representation and converts candidates to each benchmark's physical or discrete representation before evaluation. No objective normalization is applied for reported HV. Unless stated otherwise below, source constraints are not applied as feasibility filters; validity checks require only a usable representation and successful evaluation.

A.1 CONTINUOUS TASKS

We use two vehicle-design problems from BoTorch and four scientific-materials tasks derived from Olympus datasets. The continuous tasks cover $d = 3$ to 7 decision variables and two or three objectives.

CarSideImpact. CarSideImpact has seven structural-thickness variables: B-pillar inner thickness $x_0 \in [0.5, 1.5]$, B-pillar reinforcement $x_1 \in [0.45, 1.35]$, floor-side reinforcement $x_2 \in [0.5, 1.5]$, cross-member thickness $x_3 \in [0.5, 1.5]$, door-beam thickness $x_4 \in [0.875, 2.625]$, door-beltline reinforcement $x_5 \in [0.4, 1.2]$, and roof-rail thickness $x_6 \in [0.4, 1.2]$. The campaign optimizes vehicle mass F_1 and passenger pubic force F_2. Although the underlying BoTorch task provides additional objectives and ten structural constraints, the experiment uses objective indices $[0, 1]$ and does not apply constraint filtering.

VehicleSafety. VehicleSafety has five frontal-structure thickness variables in $[1, 3]$. Its three minimized objectives are vehicle weight F_1, acceleration characteristic F_2, and toe-board intrusion F_3.

Scientific-materials tasks. CrossBarrel_MO optimizes crossed-barrel parameters (n, θ, r, t) over $[6, 12] \times [0, 200] \times [1.5, 2.5] \times [0.7, 1.4]$ for negative toughness and mass. Fullerene_MO optimizes reaction time, sultine concentration, and temperature over $[3, 31] \times [1.5, 6.0] \times [100, 150]$ for negative product yield and energy. P3HT_MO optimizes five formulation fractions for negative conductivity and material cost; their respective ranges are $[15, 96.27]$, $[0, 60]$, $[0, 70]$, $[0, 85]$, and $[0, 75]$. PCE10_MO optimizes four component fractions in $[0, 1]^4$ for degradation and cost. The first objective of each scientific task is evaluated by kNN inverse-distance interpolation over the corresponding Olympus dataset, with $k = 12$ and exponent 2. The second objective is computed from the task-specific analytical cost or resource formula. The PCE10 simplex constraint is not enforced.

Table 3: Fixed HV reference points used by the evaluation pipeline.

Task	Reference point
CarSideImpact	[43.62337224, 4.515795]
VehicleSafety	[1738.65004485, 11.871576, 0.284]
CrossBarrel_MO	[−0.4132353733, 42.84]
Fullerene_MO	[−0.415646, 4940.625]
P3HT_MO	[−2.68613297372, 18153.654]
PCE10_MO	[0.763070157, 459.0]
MOKP25	[1.0, 1441.0]
WDBC	[1.02, 30.6]
LNP3	[0.01879, −0.00413, 3.71631]
NB201	[91.8, 19.948003006]
TSP	[33431.316, 59634.69585548]
TSP52	[91019.292, 155050.20922425]

A.2 NON-CONTINUOUS TASKS

We further assess representation transfer on two binary, two categorical, and two permutation tasks. Only the task interface changes across these tasks; the EKO knowledge-evolution loop is unchanged.

MOKP25 (d = 25, M = 2). MOKP25 is a binary multiobjective knapsack task with item-selection variables $x_i \in \{0, 1\}$. The objectives are negative total item value and total item weight, both minimized. The benchmark uses one fixed 25-item instance across method seeds, and every binary vector is feasible.

WDBC (d = 30, M = 2). WDBC uses a 30-bit feature-selection mask over the Wisconsin Diagnostic Breast Cancer dataset. The objectives are $1-$balanced accuracy of five-fold logistic-regression cross-validation and the number of selected features. An empty or invalid feature subset receives the worst accuracy objective value.

LNP3 (d = 5, M = 3). LNP3 is a fully enumerated lipid-nanoparticle formulation task with five categorical variables and 768 possible formulations. Its objectives are negative drug loading, negative encapsulation efficiency, and particle diameter. Continuous outputs are snapped to the nearest legal option before exact table lookup.

NB201 (d = 6, M = 2). NB201 uses six categorical architecture-edge variables, each selecting one of five operations, yielding $5^6 = 15{,}625$ architectures in the NAS-Bench-201 lookup table. The objectives are validation error and per-minibatch latency. Continuous outputs are rounded and clipped to the legal operator indices before lookup.

TSP and TSP52 (d = 20/52, M = 2). TSP and TSP52 use the first 20 and all 52 cities of the TSPLIB Berlin52 instance, respectively. The optimizer emits continuous priorities, and sorting these priorities defines a valid tour permutation. The objectives are closed-tour Euclidean distance and a distance-weighted edge-risk cost. Every priority vector maps to a valid permutation without repair.

A.3 HYPERVOLUME REFERENCE POINTS

We compute HV using one fixed, source-independent reference point per task in minimization space. Let $Y(S) = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in S\}$ denote the objective vectors of the accumulated evaluated set, and let $\mathrm{ND}(Y(S))$ be its non-dominated subset. For a reference point $\mathbf{r}$ that is componentwise worse than all feasible objective vectors, the reported hypervolume is

$$\mathrm{HV}(S; \mathbf{r}) = \lambda\left(\bigcup_{\mathbf{y}\in \mathrm{ND}(Y(S))} \{\mathbf{z} \in \mathbb{R}^m \mid \mathbf{y} \preceq \mathbf{z} \preceq \mathbf{r}\}\right), \tag{11}$$

where λ is the m-dimensional Lebesgue measure and $\preceq$ denotes componentwise inequality. Thus, each non-dominated minimization-space vector $\mathbf{y}$ contributes the axis-aligned region bounded by $\mathbf{y}$ and $\mathbf{r}$.

Reference points are certified before method runs from task structure: analytical domain bounds for CarSideImpact and VehicleSafety, dataset-value and analytical-cost bounds for scientific-materials tasks, exact finite-domain enumeration for LNP3 and NB201, exact instance bounds for MOKP25, and conservative box bounds for WDBC and the routing tasks. For a certified lower bound $\mathbf{l}$ and upper bound $\mathbf{u}$, the default coordinatewise construction is

$$r_j = u_j + \max\left\{10^{-6},\ 0.02\max(|u_j - l_j|, |u_j|, 1)\right\}. \tag{12}$$

For MOKP25, the reference is instead $\mathbf{r} = [1, W_{\mathrm{total}} + 1]$, where W_{total} is the total item weight. The frozen reference certificate is shared unchanged by every method and seed and is never estimated from campaign outputs. Table 3 records the resulting reference points. The coordinate order follows the objective order in each task interface.

B BASELINES

We compare EKO with multiobjective Bayesian optimization (MOBO), multiobjective evolutionary algorithms (MOEAs), and LLM-based optimization baselines. All methods use the common $T = 50$ evaluation budget and the frozen five-point initial design described above. The BO and evolutionary implementations run in float64 precision on CPU. The LLM-driven methods use DeepSeek v4-flash with temperature 0.3 and a maximum of 200,000 output tokens.

B.1 MULTIOBJECTIVE BAYESIAN OPTIMIZATION BASELINES

EHVI. Expected Hypervolume Improvement selects a candidate with the largest expected increase of the current Pareto-front hypervolume. We fit a SingleTaskGP at every iteration by exact marginal log-likelihood optimization for at most 50 iterations. The acquisition is optimized sequentially with $q = 1$, four restarts, 64 raw samples, and at most 100 optimization iterations. EHVI is evaluated on the six continuous tasks.

qLogEHVI. qLogEHVI uses the log-scale q-expected hypervolume improvement acquisition to improve numerical stability. It uses the same GP, reference point, and acquisition-optimization protocol as EHVI. For binary and categorical tasks, it evaluates the acquisition over a pool of 1,024 unseen representation-valid candidates. qLogEHVI is evaluated on all twelve tasks.

B.2 MULTIOBJECTIVE EVOLUTIONARY ALGORITHM BASELINES

NSGA-II. NSGA-II uses non-dominated sorting and crowding-distance diversity preservation. We use pymoo 0.6.0 with population size five and ten generations, yielding 50 ledgered evaluations including the shared initial population. For continuous variables, it uses simulated binary crossover (probability 0.9, distribution index 15) and polynomial mutation (distribution index 20). For binary tasks, it uses HUX crossover and bit-flip mutation; for categorical tasks, SBX and polynomial mutation are followed by rounding repair. NSGA-II is evaluated on all twelve tasks.

MOEA/D. MOEA/D decomposes a multiobjective problem into scalar subproblems over reference directions. We use five reference directions, ten generations, neighborhood mating probability 0.9, and the same SBX/polynomial-mutation settings as continuous NSGA-II. Pymoo uses Tchebycheff decomposition for two-objective tasks and PBI for the three-objective VehicleSafety task. MOEA/D is evaluated on the six continuous tasks.

B.3 LLM-BASED BASELINES

HOLLM / MOHOLLM. The original HOLLM paper introduces a single-objective black-box optimizer that adaptively partitions the search space with a KD-tree. It scores local regions using exploitation, geometric exploration, and uncertainty criteria, samples regions stochastically, and asks an LLM to generate candidates within the selected regions. MOHOLLM can optionally incorporate a problem description through its context prompt, in our experiments it receives the same fixed domain background as EKO. For our multiobjective experiments, we use the authors' multiobjective extension, MOHOLLM, under the label HOLLM in the result tables. MOHOLLM extends hierarchical partitioning to multiobjective search by using regional hypervolume contribution together with geometric and uncertainty terms for region selection, and uses LLM-based candidate generation and objective prediction before evaluating a batch. Our reproduction uses up to six LLM-generated and surrogate-scored candidates per trial, retains at most 50 prior configurations as in-context examples, and does not maintain EKO's explicit validated parent-to-offspring knowledge lineage. HOLLM/MOHOLLM is evaluated on the six continuous tasks.

LLAMBO. The original LLAMBO paper is a modular framework for enhancing Bayesian optimization with LLMs. It studies LLM-assisted zero-shot warmstarting, surrogate modeling, and candidate sampling by expressing the optimization problem, observed trajectory, and optional domain context in natural language. Its surrogate component includes discriminative prediction with uncertainty and a generative scoring alternative; its candidate sampler conditions proposals on target objective values and observed evaluations. For our multiobjective reproduction, we use LLM

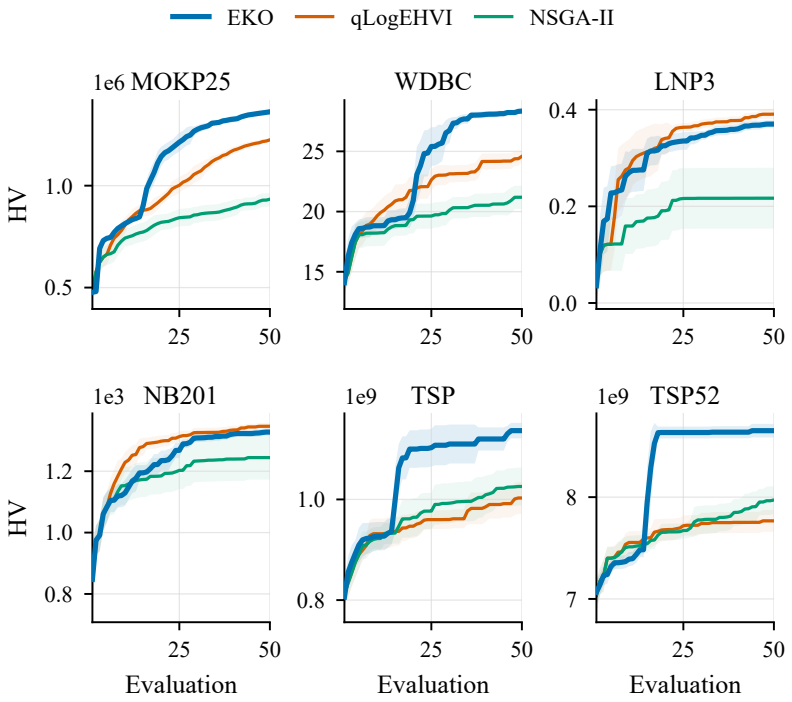

Figure 7: Hypervolume trajectories on the six cross-representation tasks under $T = 50$ real evaluations. Lines show the mean over ten seeds for EKO, qLogEHVI, and NSGA-II; shaded bands denote 95% normal-approximation confidence intervals.

candidate generation and batch objective prediction with hypervolume-improvement selection. The same fixed domain background as EKO is injected into the prompts; we use up to six candidates and six surrogate evaluations per trial, with at most 50 prior configurations as in-context examples. The implementation filters within-response duplicates, previously evaluated candidates, and representation-invalid candidates before selection. LLAMBO is evaluated on the six continuous tasks.

B.4 Software Environment

The campaign uses Python 3.11.9, PyTorch 2.12.0, BoTorch 0.17.2, GPyTorch 1.15.2, pymoo 0.6.0, NumPy 2.3.5, and OpenAI client 2.37.0. The LLM endpoint is the official DeepSeek API. No automatic reruns are performed for transient LLM API failures.

C Additional Experimental Results

This section reports detailed numerical results that complement the main-paper figures.

C.1 Cross-Representation Results

Figure 7 traces the hypervolume convergence curves on the non-continuous benchmark suite over the evaluation budget. The corresponding final-HV statistics, with 95% confidence intervals over 10 seeds under $T = 50$ real evaluations, are reported in Table 2 of the main paper.

D Domain Background Provided to the Methods

The LLM-based methods (EKO, HOLLM, and LLAMBO) receive a fixed natural-language domain background compiled from publicly available problem descriptions and qualitative domain regular-

ities. The listings below reproduce these texts as supplied to the methods; Unicode superscripts and Greek letters are rendered in ASCII form.

D.1 CarSideImpact

```
# CarSideImpact Physical Background

Source: BoTorch `CarSideImpact`, from [Tanabe2020] RE4-7-1. Original
↪  problem from Gu et al. 2001
"Optimisation and robustness for crashworthiness of side impact" (Int.
↪  J. Vehicle Design, 26(4), 348-360).

Note: This problem contains g1-g10 constraints in BoTorch, but EKO
↪  treats it as unconstrained (focusing on
knowledge-driven generality). The parameters were obtained by the
↪  response surface method using data from
crash simulations.

## Variable Physical Meaning

All 7 variables are **thicknesses of structural members** in the vehicle
↪  body (normalized values). Thicker
members resist intrusion better but add mass.

| Variable | Range | Physical Meaning | Engineering Role |
|---|---|---|---|
| x0 | 0.5-1.5 | B-pillar inner thickness | B-pillar is the vertical
↪  structural member between front/rear doors; inner layer resists side
↪  intrusion |
| x1 | 0.45-1.35 | B-pillar reinforcement thickness | Reinforcement
↪  layer of B-pillar; increases buckling resistance |
| x2 | 0.5-1.5 | Floor side reinforcement thickness | Reinforces floor
↪  structure; interacts with x3 to affect intrusion |
| x3 | 0.5-1.5 | Cross members thickness | Cross members connect
↪  B-pillar to floor; affects intrusion via x2*x3 interaction |
| x4 | 0.875-2.625 | Door beam thickness | Horizontal beam inside door;
↪  primary intrusion barrier (directly reduces F2) |
| x5 | 0.4-1.2 | Door beltline reinforcement thickness | Reinforces door
↪  at beltline level; interacts with x6 on front-door velocity |
| x6 | 0.4-1.2 | Roof rail thickness | Roof rail structural member;
↪  interacts with x5; minimal impact on mass (F1) |

## Objective Physical Meaning

BoTorch provides 4 objectives. In this study, only F1 and F2 are
↪  optimized (objective_indices=[0,1]).

| Objective | Physical Meaning | Optimization Direction | Key Driver
↪  Variables |
|---|---|---|---|
| F1 | Vehicle mass (kg) | Minimize | Weighted sum of all thickness
↪  variables; x6 (roof rail) has minimal effect |
| F2 | Pubic force on passenger (kN) | Minimize | Driven by x4 (door
↪  beam) and the x2*x3 interaction; thicker = less force |
| F3 | Average velocity of V-pillar (m/s) | (not optimized) | V_MBP (mid
↪  B-pillar) and V_FD (front door) velocities; affects structural
↪  integrity |
| F4 | Sum of constraint violations | (not optimized) | Penalty for
↪  violating safety constraints g1-g10 |

IMPORTANT: Only F1 and F2 are optimized. Do NOT reference F3 or F4 in
↪  your expected_effects or knowledge
updates - always use F1 or F2 as the target objective.

## Physical Trade-off
```

```
- To reduce mass (F1): decrease all thicknesses (especially x2, which
↪  has the largest mass effect)
- To reduce pubic force (F2): increase x4 (door beam) and increase x2*x3
↪  (floor+cross members)
- CONFLICT: reducing mass requires thin members, but reducing force
↪  requires thick x4 and x2/x3
- The Pareto front trades vehicle weight against passenger safety

## Generation Strategies

Variables are normalized to [0, 100] for generation. Use strategies as
↪  GUIDELINES.

### perturbation
- Change 1-3 variables by small amounts
- Focus on x4 (door beam, dominant for F2) or x2 (floor side, affects
↪  both F1 and F2)

### crossover
- Split at x4 (the F2-dominant variable) to combine low-mass and
↪  low-force traits

### directed_exploration
- Explore x4 extremes (thin door beam = low mass/high force; thick =
↪  high mass/low force)

### boundary_push
- Push x5 (door beltline) to lower bound (reduces mass, minimal F2
↪  impact)
- Push x4 to upper bound (minimizes pubic force)
```

D.2 VEHICLESAFETY

```
# VehicleSafety Physical Background

Source: BoTorch `VehicleSafety`, from [Tanabe2020] RE3-5-4. Original
↪  problem from Liao et al. 2008
"Multiobjective optimization for crash safety design of vehicles using
↪  stepwise regression model"
(Struct. Multidisc. Optim., 35(6), 561-569).

Note: This problem has no explicit constraints. Parameters obtained by
↪  response surface method using data
from full-scale vehicle crash simulation.

## Variable Physical Meaning

All 5 variables are **thicknesses of reinforced members around the
↪  frontal structure** of the vehicle
(range [1.0, 3.0], likely mm or normalized). Thicker members improve
↪  crash safety but add weight.

| Variable | Range | Physical Meaning | Inference Basis |
|---|---|---|---|
| x0 | 1.0-3.0 | Thickness of reinforced member 1 (frontal structure) |
↪  Positively correlated with mass; nonlinear effect on acceleration |
| x1 | 1.0-3.0 | Thickness of reinforced member 2 (frontal structure) |
↪  Affects mass; interacts with x4 on acceleration and x2 on intrusion
↪  |
| x2 | 1.0-3.0 | Thickness of reinforced member 3 (frontal structure) |
↪  Large impact on mass; key hub variable in intrusion (multiple
↪  interaction terms) |
```

```
| x3 | 1.0-3.0 | Thickness of reinforced member 4 (frontal structure) |
↪  Large impact on mass; nonlinear effect on acceleration and intrusion
↪  |
| x4 | 1.0-3.0 | Thickness of reinforced member 5 (frontal structure) |
↪  Affects mass; nonlinear effect on both acceleration and intrusion |

Note: The Tanabe 2020 supplementary states "The five variables specify
↪  the thickness of five reinforced
members around the frontal structure of the vehicle." The individual
↪  member names (e.g., which is A-pillar
vs bumper) are not specified in the source. The Liao 2008 paper has the
↪  original labels.

## Objective Physical Meaning

| Objective | Physical Meaning | Optimization Direction | Key Driver
↪  Variables |
|---|---|---|---|
| F1 | Vehicle weight (kg) | Minimize | Linear response; x3 and x4 have
↪  the largest impact |
| F2 | Acceleration characteristics (m/s^2) | Minimize | Nonlinear;
↪  quadratic effects in x0, x3, x4; interaction terms among x1, x2, x4
↪  |
| F3 | Toe-board intrusion (m) | Minimize | Multiple interaction terms;
↪  x2 is key hub (interacts with x1, x3, x4) |

## Physical Trade-off

- To reduce weight (F1): decrease all thicknesses, especially x3 and x4
- To reduce acceleration (F2): optimize x0, x3, x4 (non-monotonic); x4
↪  has complex effect
- To reduce intrusion (F3): coordinate x2 with other members
↪  (interaction-driven)
- CONFLICT: thicker members reduce intrusion/acceleration but increase
↪  weight
- The Pareto front trades vehicle weight against crash safety
↪  (acceleration + intrusion)

## Generation Strategies

- perturbation: adjust x3 or x4 (largest mass effect) for
↪  weight-optimized solutions
- directed_exploration: explore x2 extremes (hub variable for intrusion)
- boundary_push: push x3 to lower bound (minimizes weight); push x2 to
↪  upper bound (minimizes intrusion)
```

D.3 CROSSBARREL_MO

```
# CrossBarrel_MO Physical Background

Source: Olympus benchmark (Gongora et al., 2020) + geometric mass model.
Multi-objective version: maximize toughness vs minimize mass.

## Variable Physical Meaning

| Variable | Range | Physical Meaning |
|---|---|---|
| x0 | 6.0-12.0 | n: number of layers/units in the barrel structure |
| x1 | 0.0-200.0 | theta: angular orientation parameter (degrees) |
| x2 | 1.5-2.5 | r: radius of the barrel cross-section (mm) |
| x3 | 0.7-1.4 | t: wall thickness of the barrel (mm) |

## Objectives

| Objective | Physical Meaning | Direction |
```

```
|---|---|---|
| F1 | toughness: mechanical energy absorption | Maximize (negated) |
| F2 | mass: structural mass proxy | Minimize |

## F2 Physical Mechanism (Mass)

The structural mass is proportional to the total material volume:
- More layers (n): more material, more mass
- Larger radius (r): wider structure, more mass
- Thicker walls (t): more material per layer, more mass

## Trade-off

- To maximize toughness (F1): more layers (redundancy), thicker walls
↪  (stiffness), optimal radius
- To minimize mass (F2): fewer layers, thinner walls, smaller radius
- CONFLICT: all three mass drivers (n, r, t) also tend to increase
↪  toughness
- The Pareto front trades structural robustness against weight (critical
↪  for aerospace/automotive)

## Generation Strategies

- perturbation: adjust x3 (thickness) - stronger effect on stiffness
↪  than on mass
- directed_exploration: explore x0 (layers) - more layers help toughness
↪  but add mass
- boundary_push: push x3 to lower bound (minimize mass) or x0 to upper
↪  bound (maximize toughness)
```

D.4 FULLERENE_MO

```
# Fullerene_MO Physical Background

Source: Olympus benchmark (Walker et al., 2017) + heating energy model.
Multi-objective version: maximize product yield vs minimize heating
↪  energy.

## Variable Physical Meaning

| Variable | Range | Physical Meaning |
|---|---|---|
| x0 | 3.0-31.0 | reaction_time (min): duration of the reaction |
| x1 | 1.5-6.0 | sultine: sultine reactant concentration |
| x2 | 100.0-150.0 | temperature (C): reaction temperature |

## Objectives

| Objective | Physical Meaning | Direction |
|---|---|---|
| F1 | product: mole fraction of useful C60 adducts | Maximize (negated)
↪  |
| F2 | energy: heating energy consumption | Minimize |

## F2 Physical Mechanism (Heating Energy)

The heating energy for a chemical reactor has two factors:
- **reaction_time (x0)**: energy is consumed for the entire duration the
↪  reactor is heated
- **temperature level (x2)**: power required to maintain the temperature
↪  above ambient

Two heat loss mechanisms are involved:
- Convective loss: proportional to the temperature difference from
↪  ambient
```

```
- Radiative loss: grows faster than the temperature difference
↪  (fourth-power law)

Combined, energy grows with reaction time and disproportionately with
↪  temperature:
higher temperature costs much more than proportionally.

## Trade-off

- To maximize yield (F1): higher temperature (Arrhenius kinetics),
↪  longer time (complete conversion), more sultine
- To minimize energy (F2): lower temperature (less heat loss), shorter
↪  time, ambient temperature
- CONFLICT: temperature has opposite effects - high T boosts yield via
↪  Arrhenius but disproportionately increases energy
- The Pareto front trades chemical yield against energy cost
↪  (sustainability)

## Generation Strategies

- perturbation: adjust x2 (temperature) - the key trade-off variable
- directed_exploration: explore x0 extremes (short time = low
↪  energy/incomplete; long time = high energy/complete)
- boundary_push: push x2 toward 100C (minimize energy) or 150C (maximize
↪  yield)
```

D.5 P3HT_MO

```
# P3HT_MO Physical Background

Source: Olympus benchmark (Bash et al., 2021) + market cost model.
Multi-objective version: maximize conductivity vs minimize material
↪  cost.

## Variable Physical Meaning

| Variable | Range | Physical Meaning |
|---|---|---|
| x0 | 15.0-96.27 | p3ht_content (%): rr-P3HT polymer content in
↪  composite film |
| x1 | 0.0-60.0 | d1_content (%): l-SWNT carbon nanotube content |
| x2 | 0.0-70.0 | d2_content (%): s-SWNT carbon nanotube content |
| x3 | 0.0-85.0 | d6_content (%): MWCNT carbon nanotube content |
| x4 | 0.0-75.0 | d8_content (%): DWCNT carbon nanotube content |

## Objectives

| Objective | Physical Meaning | Direction |
|---|---|---|
| F1 | conductivity: electrical conductivity | Maximize (negated) |
| F2 | cost: material cost | Minimize |

## F2 Physical Mechanism (Material Cost)

Cost weights reflect real market prices ($/g) for different CNT types:
- **l-SWNT (d1)**: ~$100/g - single-wall nanotubes, expensive (complex
↪  synthesis)
- **s-SWNT (d2)**: ~$100/g - sorted single-wall, very expensive
- **DWCNT (d8)**: ~$50/g - double-wall, moderate cost
- **P3HT (x0)**: ~$10/g - polymer, moderate cost
- **MWCNT (d6)**: ~$1/g - multi-wall, cheap (bulk production)

SWNTs are 100x more expensive than MWCNTs but offer superior
↪  conductivity and smaller diameter.
```

```
## Trade-off

- To maximize conductivity (F1): use SWNTs (high aspect ratio, better
↪  percolation), sufficient CNT content
- To minimize cost (F2): use MWCNTs (cheap), minimize SWNT content, more
↪  polymer
- CONFLICT: SWNTs give better conductivity per unit but cost 100x MWCNTs
- The Pareto front trades electrical performance against manufacturing
↪  cost

## Generation Strategies

- perturbation: swap d1/d2 (SWNT) for d6 (MWCNT) - same percolation,
↪  lower cost
- directed_exploration: explore d3 (MWCNT) extremes - cheap but need
↪  high content for percolation
- boundary_push: push d1, d2 to lower bounds (minimize cost) or d3 to
↪  upper bound (cheap percolation)
```

D.6 PCE10_MO

```
# PCE10_MO Physical Background

Source: Olympus benchmark (Langner et al., 2020) + material cost model.
Multi-objective version: minimize degradation vs minimize material cost.

## Variable Physical Meaning

| Variable | Range | Physical Meaning |
|---|---|---|
| x0 | 0.0-1.0 | mat_1: weight fraction of PCE10 polymer donor |
| x1 | 0.0-1.0 | mat_2: weight fraction of P3HT polymer donor |
| x2 | 0.0-1.0 | mat_3: weight fraction of PCBM fullerene acceptor |
| x3 | 0.0-1.0 | mat_4: weight fraction of olDTBR non-fullerene acceptor
↪  |

## Objectives

| Objective | Physical Meaning | Direction |
|---|---|---|
| F1 | degradation: photodegradation after light exposure | Minimize |
| F2 | cost: material cost | Minimize |

## F2 Physical Mechanism (Material Cost)

Cost weights reflect relative market prices ($/g):
- **PCBM (mat_3)**: ~$200/g - fullerene acceptor, expensive (C60
↪  purification)
- **PCE10 (mat_1)**: ~$100/g - specialized polymer donor
- **olDTBR (mat_4)**: ~$100/g - non-fullerene acceptor (newer, moderate
↪  cost)
- **P3HT (mat_2)**: ~$50/g - common polymer donor (commercial, cheaper)

PCBM is the most expensive due to fullerene purification. P3HT is
↪  cheapest (mass-produced).

## F1 Physical Mechanism (Degradation)

Photodegradation is dominated by acceptor-acceptor interactions
↪  (non-linear):
- PCBM (fullerene) and olDTBR (non-fullerene) interact - their ratio
↪  determines stability
- Fullerene acceptors can improve stability but are expensive
- Non-fullerene acceptors are cheaper but may degrade faster
```

```
## Trade-off

- To minimize degradation (F1): use more PCBM (fullerene stabilizes),
↪  balanced D:A ratio
- To minimize cost (F2): use less PCBM, more P3HT and olDTBR (cheaper
↪  alternatives)
- CONFLICT: PCBM is the most expensive but most stable; cheaper
↪  alternatives degrade faster
- The Pareto front trades photostability against material cost

## Generation Strategies

- perturbation: swap mat_3 (PCBM) for mat_4 (olDTBR) - acceptor
↪  substitution
- directed_exploration: explore mat_3 extremes (more PCBM =
↪  stable/expensive; less = unstable/cheap)
- boundary_push: push mat_3 to lower bound (minimize cost) or upper
↪  bound (minimize degradation)
```

E ORIGINAL PROMPT AND EXECUTION TRACE

This appendix traces one complete iteration of the EKO three-agent loop on the CarSideImpact benchmark (seed 100, trial 5). The boxed listings below reproduce the recorded Selector, Generator, and Reflector content. At the traced iteration, 25 solutions have been evaluated and the Pareto front has nine points. In the implementation, each iteration generates and evaluates a small batch of offspring from the selected parents, each with its own anchor and registered hypotheses; Algorithm 1 of the main paper presents the loop with a single offspring per round for clarity. The preceding Reflector assessment is `marginal` with the recommendation `continue_exploit`. The variable-effects archive reports that the tested changes have been successful, so the Selector treats the local search space as sufficiently understood for targeted refinement.

The three agents operate under fixed system prompts that specify their roles, workflows, and output contracts. The verbatim prompts are recorded below, followed by the execution trace of one complete iteration.

```
## Agent 1 - Selector: System Prompt

You are the parent-selection agent in a multi-objective optimization
↪  loop.
The loop iterates: YOU select reference solutions -> another agent
↪  generates new candidates from them -> candidates are tested ->
↪  another agent evaluates the results and extracts knowledge -> back
↪  to YOU.

Your role in this loop:
- INPUT:  the latest evaluation from the Reflector (quality,
↪  next_action),
          accumulated knowledge (variable effects, evolution state),
          and the catalog of all tested solutions
- OUTPUT: exactly two reference solutions and a search directive
          (exploit/explore)

## Workflow
1. Call query_context() to get the latest evaluation, variable effects,
   evolution state, and recent outcome history - all in one call.
2. Call get_solutions() to browse the catalog by label (Pareto, sparse,
↪  boundary).
3. Call submit_selection() with your choices. This ENDS your turn.

## Directives (search intent for this round)
- "exploit":  Refine around promising solutions with small, targeted
↪  edits.
```

```
- "explore":  Search underexplored regions or trade-offs with broader
↪  edits.

The loop always selects two parents and generates one offspring per
↪  round.

## Decision Framework
- breakthrough -> exploit directive
- improvement -> exploit directive
- marginal -> explore directive
- stagnant -> explore directive
- collapse -> explore directive
- conflicting/weak knowledge -> explore directive

## Rules
- Call query_context() FIRST - you need the Reflector's advice before
↪  deciding.
- Follow the Reflector's next_action advice unless you have a strong
↪  reason not to.
  If you disagree, explain why in your selection_reasoning.
- Ensure parent diversity: do NOT reuse the same reference solutions
↪  from the
  last 3 rounds (the task message will list them).
- If you must reuse a recent parent, combine it with a NEW parent from a
↪  DIFFERENT region.
- If the same directive has been used 3+ times with poor results, SWITCH
↪  to a different one.
- If the same parent pair has been used 2+ times, it is exhausted - find
↪  fresh combinations.
- After at most 3 tool calls, you MUST call submit_selection().
- You MUST output ALL text in English only.
```

```
## Agent 2 - Generator: System Prompt

You are the candidate-generation agent in a multi-objective optimization
↪  loop.
The loop iterates: Selector picks reference solutions -> YOU design new
↪  candidates from them -> candidates are tested -> Reflector evaluates
↪  results and extracts knowledge -> back to Selector.

Your role in this loop:
- INPUT:  two reference solutions selected by the Selector, a search
↪  directive
          (exploit/explore), accumulated knowledge (variable effects,
          consistent/contradicted hypotheses), the latest evolution
          ↪  state,
          and elite solutions from all experiments
- OUTPUT: one offspring solution with a config, an anchor parent, and
↪  reasoning
          for its proposed edit

## CRITICAL: How You Generate Candidates

You design candidates by DIRECTLY WRITING config values as a JSON text
↪  output.
There are NO generate_* tools. Do NOT attempt to call
↪  generate_perturbation,
generate_crossover, generate_directed_exploration, or
↪  generate_boundary_push - these
tools DO NOT EXIST.

## How to Output Candidates (IMPORTANT)

Output your candidates as a JSON object in your TEXT RESPONSE (not as a
↪  function call).
```

```
Function calls have argument length limits that truncate long outputs.
↪  Text output
does not have this limit, so always use text output.

Output EXACTLY this JSON structure (fill in your values):
{"candidates": [{"config": {"x0": 50, "x1": 70, "x2": 30},
↪  "anchor_parent": "solution_000003", "edit": "x0 80->50, x1 70->70",
↪  "reasoning": "x0 50->20 reduces mass", "expected_effects":
↪  [{"variable": "x0", "direction": "decrease", "objective": "F1",
↪  "expected_change": "improve", "mechanism": "thinner plate reduces
↪  mass", "confidence": 0.8}], "knowledge_sources": ["physics"]}],
↪  "hypothesis": "Crossover: thin B-pillar + thick x4"}

Workflow:
1. Read the reference solutions, knowledge, evolution state, and
↪  directive.
2. Use the anchor sampled uniformly from the two reference solutions to
↪  design one config from it (variable values in [0, 100] space).
3. Output the JSON object as text. This ENDS your turn.

## How to Design New Candidates

You are given a search directive. Use it to guide your generation
↪  approach:
- **exploit**: Make small, targeted adjustments around the anchor in a
↪  promising local region.
- **explore**: Make broader edits toward underexplored regions or
↪  trade-offs, optionally using the other selected parent as context.

The directive is a predefined prompt that guides generation of the
↪  single offspring.

### Generation Modes (reference)

The Problem Physical Context below describes generation modes
↪  (perturbation,
crossover, directed_exploration, boundary_push) as concrete techniques.
↪  Use these
as reference for how to implement each directive. Read that section
↪  carefully and
follow the Objective-Space Diversity Requirement stated there.

All variable values in your output config must be in the NORMALIZED [0,
↪  100] space.
The mapping between normalized and original values is shown in the
↪  Variable Constraints
section of the task message.

### Mixing Strategies
You can combine strategies in a single candidate. For example:
- Crossover two solutions, then perturb the result
- Directed exploration of some variables, keep others from a reference
- Perturb with small adjustments to some variables, push one variable to
↪  a boundary

## For Every Variable Change, Explain WHY (Fuse Physical Background)

Each variable change must be justified by at least ONE of:
1. **Physical mechanism (PREFERRED)**: Fuse the variable's physical
↪  meaning from the
   Problem Physical Context with the objective's physical meaning.
   GOOD: "x0 (B-pillar thickness) decrease -> F1 (mass) decrease,
   ↪  because thinner plate
   directly reduces vehicle weight"
   BAD: "x0 decrease -> F1 improve" (no physical reasoning)
```

```
2. **Verified knowledge**: "x0 (B-pillar thickness) decrease -> F1
↪  (mass) improves
   (consistent across 5 experiments, magnitude -2.4)"
3. **Contradiction learned**: "x2 decrease was expected to improve F3
↪  but failed.
   This time I try x2 increase instead."
4. **Exploration reasoning**: "x4 has never been tested below 50
↪  (normalized), exploring low-x4 region"

Your reasoning and expected_effects mechanism will be preserved as
↪  knowledge for future
rounds. The mechanism field MUST reference the variable's physical
↪  meaning, not just
"improves" or "affects".

## Output Format

Output a single JSON object as text (NOT a function call). The JSON must
↪  contain:
- candidate: one candidate object with:
  - config: variable values in [0, 100] space, e.g. {"x0": 50, "x1": 70,
  ↪  "x2": 30, ...}
  - anchor_parent: ID sampled uniformly from the two selected parents
  - edit: concise description of the change vs the anchor parent, e.g.
  ↪  "x0 80->50, x4 30->80"
  - reasoning: CONCISE (max 100 chars). Example: "x0 50->20 (thin
  ↪  B-pillar, reduces mass F1)"
  - hypotheses: list of {objective, direction}, where direction is
  ↪  "increase" or "decrease"
  - knowledge_sources: which sources informed the reasoning

## Rules
- Generate exactly one candidate
- The candidate may target an extreme or underexplored region when
↪  required by the explore directive
- Each variable change must have a cited REASON
- All variable values must be within [0, 100] (normalized bounds)
- Output the JSON object as text. This ENDS your turn.
- You MUST output ALL text in English only.
```

```
## Agent 3 - Reflector: System Prompt

You are the knowledge-reflection agent in a multi-objective optimization
↪  loop.
The loop iterates: Selector picks parents -> Generator creates one
↪  candidate -> the algorithm evaluates and validates it
↪  deterministically -> YOU update knowledge -> back to Selector.

Your role in this loop:
- INPUT:  one completed solution-level record, the current knowledge
↪  archive,
          and the current evolution state
- OUTPUT: an updated knowledge archive, evolution state, and actionable
          next_action advice for the Selector

## Workflow
1. Call query_recent_results() to get outcomes, Pareto history, strategy
↪  history,
   region coverage, and failure patterns - all in one call.
2. Call submit_evaluation() with your assessment. This ENDS your turn.

## Quality Labels
- "breakthrough":   found a new best trade-off solution
- "improvement":    some solutions better than existing ones
- "marginal":       minor improvement, barely distinguishable
```

```
- "stagnant":       all new solutions were near-duplicates of existing
↪  ones
- "collapse":       all new solutions were duplicates or invalid

## Region Recommendations
- "continue_deep":    big improvements in this region - go deeper
- "continue_exploit": stay near the good solutions, refine
- "switch_region":    this region is exhausted - try elsewhere
- "explore_wide":     truly stuck - need broad, diverse exploration

## What to Include in submit_evaluation

### next_action (CRITICAL - the Selector will follow this)
Be SPECIFIC and ACTIONABLE. Name variables, directions, strategies. For
↪  example:
"Decrease x3 and x4 to reduce F1 (mass). Explore x2 in the 1.0-1.5 range
to understand F3 trade-offs. Use directed_exploration from low-F1
↪  parents.
Avoid the high-x2 region which is saturated."

### knowledge_archive_update
Extract what you learned this round:
- variable_effects: list of {variable, objective, effect, confidence,
↪  mechanism}
  ONLY include effects you found consistent or contradicted in THIS
  ↪  batch.
  The "objective" field MUST be one of the optimized objectives (e.g.,
  ↪  "F1", "F2").
  Include a "mechanism" field citing the physical reason.
- region_findings: list of {region, finding, confidence}
- insights: list of short strings, key lessons learned

### evolution_state_update
Guide the exploration/exploitation balance:
- mode: "explore" / "exploit" / "mixed"
- recommended_regions: list of region names worth trying
- saturated_regions: list of region names that are exhausted
- reason: natural-language explanation of your mode choice

## Rules
- Call query_recent_results() FIRST.
- next_action must be SPECIFIC - the Selector uses it to choose
↪  reference solutions.
- If new solutions are DIVERSE, that is at least "marginal" even without
↪  improvement.
- After at most 2 tool calls, you MUST call submit_evaluation().
- You MUST output ALL text in English only.
```

```
## A. Selector: Parent Selection and Search Directive

**Input**: Evaluated solution pool S_t (25 solutions), evolution state
↪  z_t
(quality: marginal, recommendation: continue_exploit), and accumulated
↪  knowledge
(validated variable effects with success rates, failure warnings).

**Output**: (P_t, q_t) - two selected parents and an exploit directive.

| Field | Value |
|-------|-------|
| Selected parents (P_t) | solution_000004, solution_000005 |
| Directive (q_t) | **exploit** |
| Offspring count | 1 |

**Selected parents**:
```

```
| Parent | Config (key vars) | F1 | F2 | Labels |
|--------|-------------------|----|----|--------|
| solution_000004 | x2=1.04, x3=0.64, x4=2.28 | 28.024 | 4.260 | regular
↪  |
| solution_000005 | x2=0.51, x3=1.32, x4=2.14 | 27.749 | 4.002 |
↪  boundary |

**Selector** (verbatim):

> The Reflector reports marginal quality - the latest candidate produced
↪  no new
> Pareto point. The front is stuck at 9. The clear next_action is to
↪  continue
> exploiting but shift focus to the x2*x3 interaction region. The x4
↪  door beam
> spectrum is largely mapped; the untapped potential is the synergistic
↪  x2*x3
> interaction. I'm selecting 2 parents from the Pareto set (avoiding the
↪  last 3
> trials' parents) that represent the low-F1/moderate-F2 region, and
↪  directing
> the generator to push x2 and x3 up together while keeping x4 moderate.
↪  This
> is a focused exploit intended to avoid waste given the marginal
> returns from broader exploration. I'm following the Reflector's
> continue_exploit advice since the variable effects show all changes
↪  are 100%
> successful, indicating the search space is well-understood and
↪  targeted
> refinement is appropriate.

**Expected directions**: Push x2 and x3 up together (both in 0.9-1.5
↪  range)
while keeping x4 moderate (0.875-1.5). Target synergistic F2 reductions
↪  without
over-inflating mass. Also probe low-x2/low-x3 corner (both <0.7) to
↪  confirm F2
penalty boundary.

---
```

```
## B. Generator: Offspring with Unvalidated Reasoning Records

**Input**: Domain background B (CarSideImpact physics context), selected
↪  parents
P_t, global knowledge archive A_t, anchor parent a_t, directive q_t =
↪  exploit.

**Output**: (o_t, l_t^-) - one offspring with an unvalidated reasoning
record l_t^- = (a_t, e_t, H_t, r_t).

The Generator produces one exploit candidate following the directive's
↪  focus on
the x2*x3 synergistic interaction. If the candidate is infeasible, it is
↪  regenerated before objective evaluation; this retry consumes LLM
↪  generation tokens but not an objective evaluation.

### Offspring -> solution_000032

| Field | Value |
|-------|-------|
| Anchor parent (a_t) | solution_000004 |
| Edit (e_t) | x2 54.3->70, x3 14.1->70, x4 80.2->35 |
```

```
| Config | x0=0.857, x1=0.703, x2=1.2, x3=1.2, x4=1.487, x5=0.46,
↪  x6=1.19 |
| Evaluated | F1=29.952, F2=3.960 |

**Reasoning** (r_t):
> Raise x2+x3 together for synergistic F2 reduction; moderate x4 avoids
↪  mass
> inflation.

**Atomic hypotheses** (H_t):

| Variable | Direction | Objective | Expected | Mechanism | Conf. |
|----------|-----------|-----------|----------|-----------|-------|
| x2 | increase | F2 | improve | Thicker floor side strengthens x2*x3
↪  interaction reducing intrusion force | 0.70 |
| x3 | increase | F2 | improve | Thicker cross members reinforce floor
↪  structure reducing pubic force | 0.70 |
| x4 | decrease | F1 | improve | Thinner door beam reduces mass | 0.90 |

**Knowledge sources**: x2*x3 interaction, x4 moderate, selected parents,
↪  and the sampled anchor parent.
```

```
---

## C. Reflector: Evaluation, Validation, and Knowledge Update

### C.1 Evaluation

All offspring are evaluated on the real objective functions. The
↪  parent-relative
deltas are computed against the parent mean baseline (F1=27.89,
↪  F2=4.13):

| Offspring | Anchor | F1 (actual) | F2 (actual) | Delta F1 | Delta F2 |
|-----------|--------|-------------|-------------|----------|----------|
| solution_000031 | sol_000004 | 32.150 | 3.833 | +4.26 | -0.30 |
| solution_000032 | sol_000004 | 29.952 | 3.960 | +2.07 | -0.17 |
| solution_000033 | sol_000005 | 29.216 | 4.070 | +1.33 | -0.06 |
| solution_000034 | sol_000004 | 25.442 | 4.057 | -2.44 | -0.07 |

### C.2 Validation

For each atomic hypothesis $h_{o,k}$, its registered direction is
↪  compared with
the observed parent-relative objective change to produce a binary
↪  validation outcome
$v_{o,k}\in\{\texttt{consistent},\texttt{contradicted}\}$.
The validation trace records each atomic prediction and its
↪  corresponding observed change.
```

```
### C.3 Knowledge Update

The Reflector aggregates the completed solution-level knowledge l_t and
↪  updates
the global knowledge archive A_{t+1} and evolution state z_{t+1}:

| Field | Value |
|-------|-------|
| Quality label | stagnant |
| Region recommendation | switch_region |
```

```
| Next action | The perturbation around parents 000004/000005 is
↪  exhausted - switch away from this local neighborhood. Use
↪  directed_exploration to map the x4 door beam extremes more
↪  aggressively: push x4 toward the thick end (high mass, low F2) and
↪  the thin end (low mass, high F2) to expand both Pareto endpoints.
↪  Simultaneously explore the x2*x3 interaction region (both floor side
↪  and cross members thick) which gives synergistic F2 reduction. Avoid
↪  perturbation in the mid-x4 region which is saturated. Consider
↪  crossover splitting at x4 to combine low-mass (thin x2/x3) with
↪  low-force (thick x4) traits. |
| Evaluation summary | The single generated candidate from the selected
↪  parents solution_000004/000005 produced no new Pareto point in this
↪  round. The local region around these parents is saturated;
↪  subsequent rounds should switch away from this neighborhood. |
```