ARCHOR: CONTRASTIVE AND SIGNATURE-GUIDED SKILL LEARNING FOR OPTIMIZATION MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) can turn natural-language optimization tasks into solver-checked answers, but current agents remain brittle under narrative rewrites and often store reusable experience at the instance level. We introduce ARCHOR, an archetype-centric skill agent that organizes experience by canonical modeling structure. ARCHOR adds two retrieval-focused modules to cluster-based skill learning. *Contrastive Archetype Representation Learning* (CARL) converts preliminary cluster memberships into weak contrastive supervision and trains a small LoRA adapter so the embedding space better respects archetype boundaries. *Signature-Guided Retrieval* (S^3R) attaches open-dimensional structural signatures to skills, filters candidates by Jaccard overlap, and then invokes semantic re-ranking only on structurally plausible skills. Across five in-distribution benchmarks, ARCHOR reaches 72.16% Micro-Avg. Pass@1, improving by 3.89 pp over the archetype-only variant and by 8.70 pp over the strongest prior skill-based baseline. It also reaches 29.15% on MIPLIB-NL and 76.34% on OOD NLCO after NANO-CO adaptation.

1 INTRODUCTION

Operational optimization underlies many industrial decision pipelines, including routing, production scheduling, energy dispatch, and workforce planning. Turning these business descriptions into executable models is still a labor-intensive expert task: practitioners must identify decision variables, formalize constraints, choose objectives, and translate the formulation into solver code. Large language models (LLMs) have therefore become an attractive interface for optimization modeling. Recent systems improve this interface through multi-agent orchestration AhmadiTeshnizi et al. (2024); Xiao et al. (2024), domain-oriented fine-tuning Jiang et al. (2025a); Huang et al. (2025a), or reinforcement learning with solver feedback Chen et al. (2026); Ding et al. (2026).

However, strong benchmark accuracy does not by itself solve the reuse problem that matters in practice. A deployed modeling agent should recognize that two differently narrated requests may instantiate the same optimization archetype, route repeated instances through a stable workflow, and extend its library when the distribution shifts. Recent experience-based systems move in this direction: AlphaOPT Kong et al. (2026) maintains solver-verified insights, while LEAN-LLM-OPT Liang et al. (2026) retrieves structured exemplars to seed workflow construction. Their stored artifacts are still largely instance-centered, so surface narratives, noisy descriptions, and missing coverage can distort retrieval even when the underlying mathematical structure is familiar.

The central difficulty is that optimization skills are defined by structure rather than by wording alone. Two assignment problems can mention different commodities, agents, and budgets while requiring the same variable pattern and constraint family; conversely, two descriptions can share terms such as "distribute" or "mix" while calling for incompatible formulations. A frozen semantic embedding and a natural-language skill card have no explicit reason to preserve these archetype boundaries, leaving robust generalization, in-distribution routing, and out-of-distribution expansion unresolved. A convincing skill-learning system should therefore show that retrieval space and selection rules align with optimization structure.

We address this issue with ARCHOR, an archetype-centric skill agent for LLM-based optimization modeling. Instead of storing one artifact per solved instance, ARCHOR clusters problems by canonical modeling structure and distills one workflow-level skill per cluster. Each skill consolidates

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.

successful and failed trajectories, including modeling paradigms and solver configurations, so that reuse targets the procedure shared by an archetype rather than the wording of a single sample. When new data arrive, the library can refine existing skills or add skills for newly discovered archetypes.

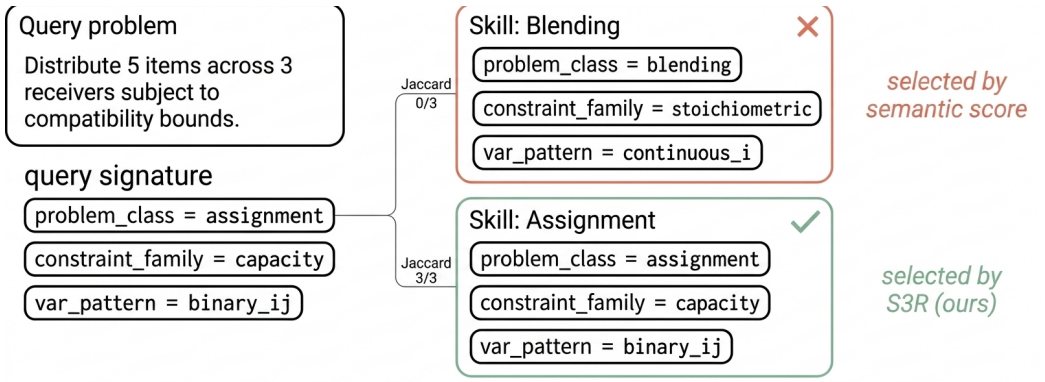

Figure 1: **Motivation.** The query and each candidate skill carry an explicit structural signature. A semantic retriever ranks the blending skill first because its narrative is close to the query, even though the signature agrees on zero of three attributes (Jaccard $0/3$). ARCHOR's Signature-Guided Retrieval instead prefers the assignment skill, whose signature matches the query on all three attributes ($3/3$), before invoking the LLM re-ranker.

Two lightweight modules make this archetype view operational. *Contrastive Archetype Representation Learning* (CARL) uses preliminary DBSCAN clusters as weak supervision: in-cluster instances form positives, cross-cluster instances form negatives, and a small LoRA adapter is trained with an InfoNCE loss to make the encoder more sensitive to archetype boundaries. *Signature-Guided Retrieval* (S^3R) attaches an open-dimensional structural signature to every skill, with attributes such as `problem_class`, `var_pattern`, and `constraint_family`. At test time, a query signature hard-filters incompatible candidates by Jaccard overlap before LLM semantic re-ranking.

Figure 1 illustrates why both components are needed. For a query about distributing items across receivers, a semantics-only selector ranks a blending skill above the correct assignment skill because the surface wording overlaps. ARCHOR instead extracts a structural signature, removes incompatible candidates before re-ranking, and selects the assignment skill. CARL improves the embedding used to form and retrieve archetype clusters, while S^3R provides a discrete guardrail against semantically plausible but structurally wrong skills. Appendix **??** further reports KNN diagnostics on NANO-CO (Appendix **??**).

Empirically, ARCHOR reaches a Micro-Avg. accuracy of *72.16%* on five in-distribution benchmarks, improving by *8.70 pp* over the strongest prior skill-based baseline (Trace2Skill) and by *3.89 pp* over the archetype-only variant of our system. On MIPLIB-NL, it reaches *29.15%*, exceeding DeepSeek-V3.2-Thinking by *6.77 pp*; after NANO-CO adaptation, it reaches *76.34%* on OOD NLCO. Our contributions are threefold:

1. We formalize an archetype-centric view of skill learning for optimization modeling, in which reusable experience is organized by the canonical variable, constraint, and objective pattern of a problem rather than by its surface narrative or instance identity.
2. We introduce two complementary modules that operationalize this view. CARL turns pipeline-produced cluster memberships into weak contrastive supervision and trains a lightweight LoRA adapter so the retrieval embedding respects archetype boundaries. S^3R attaches an open-dimensional structural signature to every skill and hard-filters candidates by Jaccard overlap before semantic re-ranking, so structurally incompatible skills are excluded before the LLM ever sees them.
3. We evaluate ARCHOR across five in-distribution benchmarks, the industrial-scale MIPLIB-NL, and the OOD NLCO benchmark; ablation, KNN retrieval diagnostics, DB-

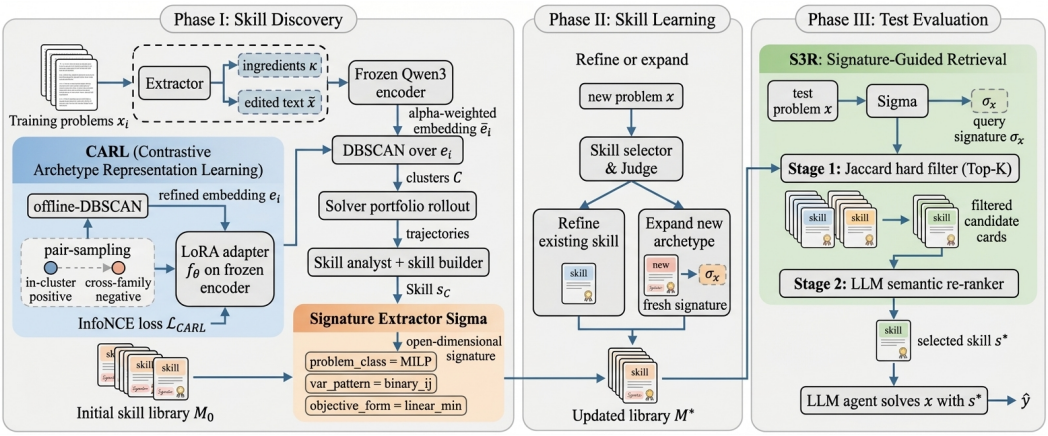

Figure 2: **Overview of ARCHOR. Phase I** builds the initial skill library $\mathcal{M}_0$: an extractor produces ingredients κ_i and edited text $\tilde{x}_i$, a frozen Qwen3 encoder gives an α-weighted embedding $\bar{e}_i$, *CARL* fine-tunes a LoRA adapter with InfoNCE using in-cluster positives and cross-family negatives to yield the refined embedding e_i, DBSCAN over e_i groups problems by archetype, and a skill analyst + skill builder distill each cluster into a skill s_C to which the *Signature Extractor* Σ attaches an open-dimensional structural signature. **Phase II** refines existing skills or expands the library with new-archetype skills. **Phase III** performs *Signature-Guided Retrieval* (S^3R): Σ produces the query signature σ_x, stage-1 Jaccard hard-filtering keeps top-K candidates, stage-2 LLM semantic re-ranking selects a single skill s^* that guides the LLM agent.

SCAN sensitivity, and coverage analyses jointly show that both modules contribute independently and that the resulting skill library remains transferable under distribution shift.

2 Related Work

2.1 LLMs for Modeling and Solving Optimization Problems

Post-Training Methods. A first line of work embeds optimization competence directly into the model weights. Recent representatives cover supervised fine-tuning (ORLM Huang et al. (2025a), OptMATH Lu et al. (2025), AutoOR Motwani et al. (2026)), preference optimization (LLMOPT Jiang et al. (2025a)), and reinforcement learning with solver-based reward (SIRL Chen et al. (2026), OR-R1 Ding et al. (2026)). These approaches trade training cost for model-internal knowledge and remain the strongest option when the target problem distribution is fixed in advance.

Agent-Based Methods. A second line keeps the base model frozen and instead composes a multi-stage LLM pipeline in which formulation, coding, execution, and repair are handled by distinct agents. Chain-of-Experts Xiao et al. (2024) and OptiMUS AhmadiTeshnizi et al. (2024) route subtasks through role-specialized experts glued together by a conductor. ORMind Wang et al. (2025) and ORThought Yang et al. (2025b) layer cognitive-style heuristics on top of the pipeline so that business intent is progressively refined into a mathematical formulation and executable solver code.

Experience-Enhanced Methods. A third line goes further and lets the agent accumulate its own reusable artifacts. DRoC Jiang et al. (2025b) and ConstraintLLM Shi et al. (2025) shred natural-language constraints into retrievable atomic units for formal reasoning. MIRROR Shi et al. (2026) runs hierarchical retrieval over exemplar libraries to seed task-specific formulations and code. LEAN-LLM-OPT Liang et al. (2026) indexes structured reference examples that anchor its workflow synthesis; AlphaOPT Kong et al. (2026) converts failures into structured, applicability-tagged insights and iteratively refines their scope.

2.2 Skill Learning in LLM Agents

Compared to task-specific experience augmentation, skill-equipped agents pursue a more transferable form of knowledge, each skill is meant to be reusable across an entire class of tasks, not tuned to one instance. XSkill Jiang et al. (2026a) disentangles operational traces from the underlying structural skill. EvoSkill Alzubi et al. (2026), Memento-Skills Zhou et al. (2026), and AutoSkill Yang et al. (2026) focus on automatic discovery, routing, and continual evolution of the skill catalog. Trace2Skill Ni et al. (2026) folds trajectory-local lessons into a unified conflict-free directory; Mem-Skill Zhang et al. (2026) treats memory operations themselves as trainable skills. Across all of these, however, the encoder that supports skill retrieval stays frozen and every skill is indexed exclusively by its natural-language name and description. This leaves two structural gaps: the embedding space is never told what an archetype is, and a semantics-only retriever loses well-formed skills to more generic-sounding neighbours. ARCHOR closes both gaps: it learns a retrieval embedding directly from task-driven contrastive supervision, and it attaches an open-dimensional structural signature to every skill so that retrieval performs a discrete pre-filter before semantic re-ranking.

3 Methodology

3.1 Preliminaries

Optimization Problem Archetype. An optimization *archetype* $a \in \mathcal{A}$ is the modeling pattern that remains after scenario names and business context have been stripped away: the abstract variable roles, objective template, and constraint relations. We write a narrated instance as $x = g_u(a)$, where u supplies the application setting and g_u renders the abstract pattern as natural language. The intended invariance is $f_{\text{model}}(a) = f_{\text{model}}(g_u(a))$ for any valid modeling routine f_{model}. This equality is an idealization, but it captures the property we want the agent to exploit: problems that instantiate the same a should call for the same variable types, constraint families, objective structure, and parameter semantics even when their stories look unrelated.

Skills. A *skill* $s \in \mathcal{S}$ is the reusable procedural unit attached to an archetype family. We represent it as $s = (m, w, p)$, where m records the skill name and applicability boundary, w gives the modeling and solving recipe from variable declaration through result parsing, and p records recurring traps observed in rollouts for that family. Thus a skill is neither a single demonstration nor a free-form memory item: it is a compact guide intended to steer future instances with the same structural pattern. Appendix **??** gives an example.

3.2 An Overview of ArchOR

Figure 2 summarizes the full system. In **Phase I**, ARCHOR converts each training problem into structural ingredients and a scenario-neutral rewrite, embeds the two views, applies CARL to make the representation more archetype-aware, clusters the refined vectors with DBSCAN, and distills each cluster into one workflow-level skill. The same phase also runs the Signature Extractor so that every skill receives a machine-comparable structural signature. In **Phase II**, new training evidence is used either to revise an applicable skill or, when no skill passes the applicability judge, to create a new library entry after a successful rollout. In **Phase III**, S^3R first narrows the candidate set by signature overlap and then lets the LLM selector choose the final skill for inference.

3.3 Phase I: Skill Discovery via Cluster-Based Distillation

Problem Archetype Representation. For each training instance x_i, the first step is to construct a vector view that tracks a_i rather than the surface story. The extractor therefore produces two complementary artifacts: an ingredients JSON κ_i that names abstract variables, objective slots, and constraint roles, and a rewritten statement $\tilde{x}_i$ in which domain-specific entities are replaced with placeholders while the quantities and logical relations are retained. We deliberately treat κ_i as a regenerated structural description, not as copied text from the original instance.

The initial archetype embedding is a normalised convex mixture, $\bar{e}_i = \text{Norm}(\alpha \cdot w_i + (1-\alpha) \cdot v_i)$, where w_i and v_i come from Qwen3-Embedding-v3 applied to κ_i and $\tilde{x}_i$ respectively, and $\alpha \in [0, 1]$

trades off the two sources. Since Qwen3 is a general-purpose encoder that has never been told what an "archetype" is, $\bar{e}_i$ inherits the encoder's blind spots: instances of different archetypes that share surface vocabulary can end up close in embedding space, while paraphrases of the same archetype can be pushed apart. DBSCAN over $\bar{e}_i$ therefore reveals archetype boundaries only approximately.

Contrastive Archetype Representation Learning (CARL). CARL fixes this by making the encoder learn its own archetype boundary from signals the pipeline already produces. Concretely, we run an offline DBSCAN pass over $\{\bar{e}_i\}$ to obtain preliminary clusters $\bar{\mathcal{C}}$. For each anchor x_i, we sample one positive x_i^+ from the same cluster in $\bar{\mathcal{C}}$ and one negative x_i^- from a different cluster, preferring negatives drawn from clusters solved by a disjoint solver family, since a cross-solver mismatch is stronger evidence of a structural gap. Let f_θ be a LoRA adapter attached to the frozen Qwen3 encoder and define $e_i = \mathrm{Norm}(f_\theta(\bar{e}_i))$. We fit θ with an InfoNCE loss

$$\mathcal{L}_{\mathrm{CARL}} = -\sum_i \log \frac{\exp(\mathrm{sim}(e_i, e_i^+)/\tau)}{\sum_{j \in \mathcal{N}_i \cup \{i^+\}} \exp(\mathrm{sim}(e_i, e_j)/\tau)}, \tag{1}$$

with cosine similarity sim, temperature τ, and in-batch negatives $\mathcal{N}_i$. Only θ ($\sim$3.6M parameters) is updated; Qwen3 stays frozen. This is bootstrapped rather than circular: the initial KNN Hit@1 in Table **??** is 0.820, i.e., the pre-CARL clusters already carry a strong signal, and InfoNCE amplifies it via cross-solver-family negatives. CARL adds no annotation and no extra LLM calls; we simply swap $\bar{e}_i$ for e_i at inference.

Solver Portfolio Rollout. Skill construction needs examples of what works and what fails, so ARCHOR evaluates a small portfolio instead of committing to one formulation route. Given $(\tilde{x}_i, \kappa_i)$, a solver selector ranks the catalog $\mathcal{B}$ and returns $\mathcal{B}_i$, the top-k backends or modeling paradigms for the instance. For every $b \in \mathcal{B}_i$, the LLM writes a formulation z_i^b, generates executable code $c_{i,t}^b$, observes runtime feedback $o_{i,t}^b$, and iterates until the budget is exhausted or a verified answer is obtained.

After T_i^b code iterations, we obtain a feasible answer $\hat{y}_i^b$ and record the trajectory

$$\tau_i^b = \big(z_i^b, c_{i,1}^b, o_{i,1}^b, c_{i,2}^b, o_{i,2}^b, \ldots, c_{i,T_i^b}^b, \hat{y}_i^b\big). \tag{2}$$

Trajectories in $\mathcal{T}_i = \{\tau_i^b \mid b \in \mathcal{B}_i\}$ are labelled positive when $\hat{y}_i^b = y_i^*$ and negative otherwise, giving the split $\mathcal{T}_i^+$ / $\mathcal{T}_i^-$.

Trajectory Analysis. The skill analyst A_η converts the raw rollouts into structured notes before any cluster-level fusion is attempted. Successful trajectories contribute reusable SOPs about formulation choices, solver settings, validation checks, and output parsing; failed trajectories contribute pitfall notes that record the concrete modeling or execution cause:

$$q_i = A_\eta(\mathcal{T}_i^+, \mathcal{T}_i^-) = (r_i, p_i). \tag{3}$$

Each q_i pairs an SOP set r_i with a pitfall set p_i and serves as one atomic unit that will later be fused into a skill.

Cluster-Based Skill Distillation. After representation learning, each instance is associated with e_i, rollout records $\mathcal{T}_i$, and analysis notes q_i. We cluster the normalized embeddings with DBSCAN Ester et al. (1996), $\mathcal{C} = \mathrm{DBSCAN}(\{e_i\}_{i=1}^N; \epsilon, m)$, using cosine distance; ϵ controls the neighborhood radius and m the core-point threshold. A cluster C is treated as a candidate archetype group.

The skill builder then summarizes all analyses in C into $s_C = B_\eta(\{q_i \mid i \in C\})$, and $\mathcal{M}_0 = \{s_C \mid C \in \mathcal{C}\}$ becomes the starting library. Since the inputs to a cluster usually differ in narrative but agree in modeling structure, the builder can retain the common procedure while discarding scenario-specific accidents.

Skill Signature Extraction. Cluster-based distillation improves skill quality, but the resulting skills are still stored as free-form natural-language documents and retrieved purely via LLM-based semantic matching over names and descriptions. This has two drawbacks: (i) semantically similar but structurally distinct skills confuse the selector, and (ii) many well-formed skills are never

Table 1: Pass@1 solving accuracy across five benchmarks. ARCHOR-D uses DeepSeek-V3.2 as the backbone, whereas ARCHOR-Q uses Qwen3-235B-A22b-instruct-2507. We mark the best, second-best, and third-best entries with **bold**, underline, and *italic underline*, respectively.

Category	Models / Methods	Macro-Avg.	Micro-Avg.	Rank	OptiBench	Mamo.C	OptMATH	IndustryOR	ComplexOR
General Models	GPT-5.4	51.71	57.82	9	69.09	47.39	46.39	29.00	66.67
	Gemini-3.1-Pro	53.58	57.88	8	66.56	46.45	54.22	34.00	66.67
	Qwen3-235B	42.63	52.36	10	63.80	41.71	39.76	29.00	38.89
	DeepSeek-V3.2	43.21	49.45	12	62.64	27.49	40.36	30.00	55.56
Agent-Based Methods	Chain-of-Experts	38.84	52.36	11	62.31	51.66	35.54	28.00	16.67
	OptiMUS	34.14	45.18	16	60.99	27.49	22.89	26.00	33.33
	ORMind	36.69	47.37	14	61.82	34.60	21.69	32.00	33.33
	ORThought	41.92	46.91	15	57.69	38.86	26.51	31.00	55.56
Experience-Enhanced Methods	LEAN-LLM-OPT	41.23	49.27	13	62.15	43.13	22.89	28.0	50.0
	AlphaOPT	50.92	58.55	7	70.25	54.50	36.75	32.00	61.11
Skill-Based Methods	Trace2Skill	56.97	63.46	5	75.21	54.03	49.40	34.00	72.22
	ArchOR-Q	54.64	61.46	6	71.74	53.55	54.22	27.00	66.67
	ArchOR-D	62.04	68.27	4	77.02	63.51	61.45	36.00	72.22
	+ CARL	*62.81*	*69.90*	*3*	*78.50*	*64.85*	*63.00*	*36.50*	72.22
	+ S^3R	64.63	70.90	2	79.20	65.30	63.85	37.00	**77.78**
	+ CARL + S^3R	**65.83**	**72.16**	**1**	**81.15**	**66.40**	**65.80**	**38.00**	**77.78**

invoked because they lose to more generic-sounding neighbors during retrieval. We address both by attaching a structured *signature* to every skill. Concretely, a dedicated Signature Extractor Σ, an LLM prompted with the skill document and the trajectory analyses in its source cluster, emits a JSON object

$$\begin{aligned} \sigma_s &= \Sigma(s_C, \{q_i \mid i \in C\}), \\ \sigma_s &= \{(k_1, V_1), \ldots, (k_L, V_L)\}, \end{aligned} \tag{4}$$

where each (k_ℓ, V_ℓ) is an attribute name and a set of canonical tokens. The schema is intentionally *open-dimensional*: Σ freely chooses the attributes it needs (e.g., `problem_class`, `var_pattern`, `constraint_family`, `objective_form`, `solver_backend`) and returns only those it is confident about. This preserves the flexibility of a natural-language catalog while grounding each entry in a discrete, machine-comparable signature. The same Σ is applied to every test problem to produce a query signature σ_x, adding at most one LLM call per test instance, which is amortized with the extractor call already required in Phase III.

3.4 PHASE II: SKILL LEARNING

After $\mathcal{M}_0$ has been built, later training instances are processed as library-maintenance events. The extractor again returns $(\kappa, \tilde{x})$, the selector proposes a skill from the current catalog, and a judge checks whether the proposed applicability boundary really covers the new problem. Accepted matches trigger skill refinement with the new trajectory as evidence; rejected matches are queued as possible new archetypes and only become library entries after at least one successful rollout.

Skill Refinement. Suppose the selected skill is $s^\star = (m^\star, w^\star, p^\star)$. We run the agent with $s^\star$ as guidance and store the trajectory τ. If the final answer is correct, A_η extracts additional SOP evidence to strengthen $w^\star$; otherwise it extracts a diagnosed failure mode for $p^\star$. The refiner incorporates the resulting analysis q while keeping the skill name and applicability boundary stable, producing s_{refined} as a replacement for $s^\star$.

Skill Expansion. When the judge finds no applicable skill, the instance is treated as an uncovered archetype candidate. ARCHOR repeats the Phase-I portfolio rollout to obtain $\mathcal{T}_x = \{\tau_x^b \mid b \in \mathcal{B}_x\}$, analyzes the trajectories, and creates $s_{\text{new}} = B_\eta(\{q\})$ only if some rollout is verified correct. The update $\mathcal{M}_{t+1} = \mathcal{M}_t \cup \{s_{\text{new}}\}$ is therefore gated by evidence of a working procedure, which prevents the library from storing unsupported advice.

3.5 PHASE III: TEST EVALUATION WITH TWO-STAGE RETRIEVAL

At test time, the learned library $\mathcal{M}^*$ is frozen. For a query x, the extractor again produces κ and $\tilde{x}$, and retrieval proceeds through *Signature-Guided Retrieval* (S^3R).

Stage 1: Signature-Based Hard Filtering. We apply Σ to the test problem to obtain σ_x. For every skill $s \in \mathcal{M}^*$ with signature σ_s, we compute a weighted Jaccard similarity over shared attributes,

$$J(\sigma_x, \sigma_s) = \sum_{k \in \sigma_x \cap \sigma_s} \omega_k \cdot \frac{|V_k^x \cap V_k^s|}{|V_k^x \cup V_k^s|}, \tag{5}$$

where $\omega_k = 1/|\sigma_x \cap \sigma_s|$ by default and attributes present in only one signature contribute zero. Because the schema is open-dimensional, the sum is naturally taken over the intersection of attributes. We keep the top-K skills as candidates ($K = 8$); when fewer than K skills yield $J > 0$ (e.g., Σ emits an unusual attribute set for the query), we fall back to including the highest-J skills regardless of overlap so that stage 2 is never starved of candidates.

Stage 2: Semantic Re-ranking. The LLM-based skill selector from Phase II is then applied only over the K candidates, using their names and descriptions as retrieval cues; the selected skill guides inference and the prediction is finally evaluated against y^*. The two-stage design keeps the flexibility of natural-language matching while excluding the semantically similar but structurally incompatible skills that were the main source of degradation in the flat retriever.

4 EXPERIMENTS

Datasets for Optimization Skills Construction. The skill-building split is sampled from OptMATH-Train Lu et al. (2025). We select 300 instances across 20 scenarios, with class counts of 75 LP, 75 MILP, 75 IP, 45 NLP, and 30 SOCP problems. The first half is used to discover the initial cluster-level library; the second half is reserved for incremental learning, where skills can be revised or added. Appendix **??** gives the remaining configuration.

Evaluation Benchmarks. The in-distribution evaluation uses five benchmark suites: OptiBench Yang et al. (2025c), OptMATH-Bench Lu et al. (2025), Mamo.C Huang et al. (2025b), IndustryOR Huang et al. (2025a), and ComplexOR Xiao et al. (2024). We additionally report transfer to the hard split of NLCO Jiang et al. (2026b). Because these datasets vary in description length, formulation family, and evaluation protocol, they test both numerical solving accuracy and the ability to route across archetypes. Dataset details appear in Appendix **??**.

Baselines. The comparison covers four groups. General LLM baselines (GPT-5.4 OpenAI (2026), Gemini-3.1-Pro Google DeepMind (2026), DeepSeek-V3.2 DeepSeek-AI (2025), and Qwen3-235B Yang et al. (2025a)) are prompted to write solver code directly. Agent frameworks include CoE Xiao et al. (2024), OptiMUS AhmadiTeshnizi et al. (2024), ORMind Wang et al. (2025), and ORThought Yang et al. (2025b). Experience-library methods include AlphaOPT Kong et al. (2026) and LEAN-LLM-OPT Liang et al. (2026). The skill-based baseline is Trace2Skill Ni et al. (2026). Reproduction choices are documented in Appendix **??**.

The experiments are organized around six questions:

Q1: How does the optimization modeling and solving performance of ARCHOR compare with baseline methods?

Q2: How do skills enhance the agent's capability in modeling and solving optimization problems?

Q3: What advantages does the clustering strategy offer for skill utilization?

Q4: How well does ARCHOR generalize to in-distribution and out-of-distribution domains?

Q5: How effective is ARCHOR in handling real-world and large-scale optimization problems?

Q6: What are the individual and joint contributions of CARL and S^{3}R to archetype-centric skill reuse?

4.1 MAIN RESULTS

Table 1 reports Pass@1 solving accuracy (SA) for all compared systems.

Table 2: Ablation study of ARCHOR on the SA metric across 5 benchmarks. A checkmark indicates the module is enabled. CARL: Contrastive Archetype Representation Learning; S^3R: Signature-Guided Retrieval.

Experimental Groups	Tool-Call Agent	Skill Distillation	Cluster-Based Distillation	Skill Learning	CARL	S^3R	**OptiBench**	**Mamo.C**	**OptMATH**	**IndustryOR**	**ComplexOR**
Group 1	☑	☐	☐	☐	☐	☐	73.22	53.08	54.21	34.00	66.67
Group 2	☑	☑	☐	☐	☐	☐	69.75	56.40	59.64	32.00	66.67
Group 3	☑	☑	☑	☐	☐	☐	74.71	59.24	60.84	35.00	72.22
Group 4	☑	☑	☑	☑	☐	☐	77.02	63.51	61.45	36.00	72.22
Group 5	☑	☑	☑	☑	☑	☐	78.50	64.85	63.00	36.50	72.22
Group 6	☑	☑	☑	☑	☐	☑	79.20	65.30	63.85	37.00	77.78
Group 7	☑	☑	☑	☑	☑	☑	**81.15**	**66.40**	**65.80**	**38.00**	**77.78**

Comparison of ARCHOR and Baseline Methods (To Q1). Table 1 shows that ARCHOR (DeepSeek-V3.2 backbone) consistently achieves state-of-the-art performance. The archetype-only ARCHOR-D already attains a Micro-Avg. of 68.27%; enabling CARL brings it to 69.90%, adding S^3R further to 70.90%, and the full system reaches **72.16%**, an 8.70 pp gain over the strongest prior baseline Trace2Skill (63.46%) and a 3.89 pp gain over the archetype-only variant. To isolate the effect of the underlying LLM, we also run inference with Qwen3-235B-A22b-instruct-2507 (ARCHOR-Q): it reaches 61.46%, comparable to Trace2Skill and above most baselines, confirming that gains stem from the framework rather than the backbone. On the harder OptMATH and Mamo.C benchmarks, our advantage is especially pronounced.

4.2 ABLATION STUDY

The Contribution of Skills to Agent Performance (To Q2). Table 2 reports Pass@1 under each ablation. Adding one skill per training sample (Group 2) is not enough: 81 of 129 entries (62.8%) are never selected, and many descriptions encode instance-specific applicability rather than reusable modeling logic. This explains why the gain over the tool-call agent remains limited.

Enhancement of Skill Application via Clustering Strategies (To Q3). Group 3 replaces per-instance storage with cluster-level distillation and improves across the benchmark suite. The resulting library has 46 entries, only 7 of which (15.2%) are unused, compared with 62.8% unused entries in the per-sample library. The reduction indicates that archetype clustering makes the catalog easier to route and less redundant. Appendix **??** further compares this representation against raw problem-text embeddings.

Skill Learning on In-Distribution Data (To Q4-1). Group 4 of Table 2 shows improvements over Group 3 across almost all benchmarks; exceptions are IndustryOR and ComplexOR, where high error rates Xiao et al. (2025) and small sample sizes weaken statistical significance. To trace skill-library growth, we take checkpoints at 46/49/53/56 skills on OptiBench: accuracy trends upward (74.71%→74.05%→75.37%→77.02%), with a minor dip at 49, indicating that expanding coverage within the same distribution yields sustained gains (bar chart in Appendix **??**).

4.3 DISCUSSIONS

Discussion. The ablations indicate that representation learning and structural filtering address complementary stages of retrieval: CARL improves which skills are nearby, while S^3R prevents structurally incompatible skills from reaching the final selector. The remaining errors concentrate on incomplete extracted signatures, suggesting that lightweight symbolic validation is a promising next step.

5 CONCLUSION

We propose ARCHOR, an archetype-centric skill system with two lightweight modules: CARL uses pipeline-produced cluster memberships as weak supervision to fine-tune a LoRA adapter on the embedding, and S^3R attaches open-dimensional structural signatures to skills and hard-filters candidates by Jaccard similarity before semantic re-ranking. Across five ID benchmarks, MIPLIB-

NL, and OOD NLCO, they jointly deliver +3.89 pp Micro-Avg. and cut never-invoked skills from 15.2% to 4.3% without human annotation.

REFERENCES

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In *Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria, 2024.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems, 2026.

Yitian Chen, Jingfan Xia, Siyu Shao, Dongdong Ge, and Yinyu Ye. Solver-informed RL: Grounding large language models for authentic optimization modeling. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*, 2026.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models, 2025.

Zezhen Ding, Zhen Tan, Jiheng Zhang, and Tianlong Chen. OR-R1: Automating modeling and solving of operations research optimization problem via test-time reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 228–236, 2026.

Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231. AAAI Press, 1996.

Google DeepMind. Gemini 3.1 pro. `https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview`, 2026.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. ORLM: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 2025a.

Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. LLMs for mathematical modeling: Towards bridging the gap between natural and mathematical languages. In *Findings of the Association for Computational Linguistics 2025*, pp. 2678–2710, 2025b.

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. LLMOPT: Learning to define and solve general optimization problems from scratch. In *Proceedings of the 13th International Conference on Learning Representations*, 2025a.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. XSkill: Continual learning from experience and skills in multimodal agents, 2026a.

Xia Jiang, Yaoxin Wu, Chenhao Zhang, and Yingqian Zhang. DRoC: Elevating large language models for complex vehicle routing via decomposed retrieval of constraints. In *Proceedings of the 13th International Conference on Learning Representations*, 2025b.

Xia Jiang, Jing Chen, Cong Zhang, Jie Gao, Chengpeng Hu, Chenhao Zhang, Yaoxin Wu, and Yingqian Zhang. Reasoning in a combinatorial and constrained world: Benchmarking LLMs on natural-language combinatorial optimization, 2026b.

Minwei Kong, Ao Qu, Xiaotong Guo, Wenbin Ouyang, Chonghe Jiang, Han Zheng, Yining Ma, Dingyi Zhuang, Yuhan Tang, Junyi Li, Shenhao Wang, Haris Koutsopoulos, Hai Wang, Cathy Wu, and Jinhua Zhao. AlphaOPT: Formulating optimization programs with self-improving LLM experience library, 2026.

Kuo Liang, Yuhang Lu, Jianming Mao, Shuyi Sun, Chunwei Yang, Congcong Zeng, Xiao Jin, Hanzhang Qin, Ruihao Zhu, and Chung-Piaw Teo. Large-scale optimization model autoformulation: Harnessing LLM flexibility via structured workflow, 2026.

Hongliang Lu, Zhonglin Xie, Yaoyu Wu, Can Ren, Yuxuan Chen, and Zaiwen Wen. OptMATH: A scalable bidirectional data synthesis framework for optimization modeling. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Sumeet Ramesh Motwani, Chuan Du, Aleksander Petrov, Christopher Davis, Philip Torr, Antonio Papania-Davis, and Weishi Yan. AutoOR: Scalably post-training LLMs to autoformalize operations research problems, 2026.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills, 2026.

OpenAI. Introducing GPT-5.4. `https://openai.com/index/introducing-gpt-5-4/`, 2026.

Weichun Shi, Minghao Liu, Wanting Zhang, Langchen Shi, Fuqi Jia, Feifei Ma, and Jian Zhang. ConstraintLLM: A neuro-symbolic framework for industrial-level constraint programming. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15999–16019, 2025.

Yifan Shi, Jialong Shi, Jiayi Wang, Ye Fan, and Jianyong Sun. MIRROR: A multi-agent framework with iterative adaptive revision and hierarchical retrieval for optimization modeling in operations research, 2026.

Zhiyuan Wang, Bokui Chen, Yinya Huang, Qingxing Cao, Ming He, Jianping Fan, and Xiaodan Liang. ORMind: A cognitive-inspired end-to-end reasoning framework for operations research. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Industry Track)*, pp. 104–131, 2025.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Ziyang Xiao, Jingrong Xie, Lilin Xu, Shisi Guan, Jingyan Zhu, Xiongwei Han, Xiaojin Fu, WingYin Yu, Han Wu, Wei Shi, Qingcan Kang, Jiahui Duan, Tao Zhong, Mingxuan Yuan, Jia Zeng, Yuan Wang, Gang Chen, and Dongxiang Zhang. A survey of optimization modeling meets LLMs: Progress and future directions. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025a.

Beinuo Yang, Qishen Zhou, Junyi Li, Chenxing Su, Panagiotis Angeloudis, and Simon Hu. OR-Thought: Benchmarking and automating logistics optimization modeling, 2025b.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. AutoSkill: Experience-driven lifelong learning via skill self-evolution, 2026.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. OptiBench meets ReSocratic: Measure and improve LLMs for optimization modeling. In *Proceedings of the 13th International Conference on Learning Representations*, 2025c.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. MemSkill: Learning and evolving memory skills for self-evolving agents, 2026.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-skills: Let agents design agents, 2026.