SEMANTIC RELAXATION WITH LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, agentic workflows have achieved substantial advances across downstream tasks by enabling multiple Large Language Models to interact with external environments (*e.g.*, tools, memory, and API). Nevertheless, their *discrete* and *non-differentiable* nature poses a fundamental optimization barrier in that external feedback often fails to translate into explicit update directions. In this work, we carefully design **SemanticRelaxation**, an agentic optimization framework that semantically bridges discrete generation with continuous optimization. Specifically, SemanticRelaxation integrates two core components: (i) *Semantic Representation Learning* (SRL), which learns a joint embedding space by promoting cross-modal correlation between descriptions and their corresponding artifacts while preserving semantic structure; and (ii) *Barycentric Semantic Optimization* (BSO), which explicitly guides semantic directions through barycentric aggregation and accumulates historical updates as trajectory momentum after parallel transport. Finally, SemanticRelaxation effectively encourages agentic workflows by grounding executable artifact refinements in retrieved semantic trajectories within an iterative optimization loop. Our strategy achieves state-of-the-art performance on benchmarks spanning Agentic Workflows, Code Generation, Molecular Editing, and Neural Architecture Search, increasing the success rate on M3Tool by up to 12.2% and reducing token consumption on HumanEval by 29.2%. Comprehensive empirical analyses further support the effectiveness and validity of our framework.

1 INTRODUCTION

Recent advances in Large Language Models (LLMs) have enabled the development of agentic workflow, where specialized agents collaborate and use external tools to tackle complex tasks. Especially, existing approaches automate a broad range of domains, including web navigation, embodied interaction, planning, and code generation (Yao et al., 2022; Wang et al., 2022; Shridhar et al., 2021; Ma et al., 2024; Chen et al., 2021a; Austin et al., 2021), while reducing dependence on manual design required to understand tasks, implement procedures, and coordinate execution.

First, Chen et al. (2024a) develop agents and workflows from natural-language requirements, while Hu et al. (2025) generate new agent implementations from an archive of evaluated designs. Zhang et al. (2025b) search executable workflows using Monte Carlo tree search, and Shang et al. (2025) evolve and recombines functional modules . Li et al. (2026) guide hierarchical search with a learned value model, whereas Zhang et al. (2025a) jointly optimizes a distribution over agent architectures and their constituent operators. However, these methods inherently struggle to turn execution outcomes into concrete improvements, mainly due to discrete decisions and non-differentiable operations.

More broadly, progress in AI has repeatedly relied on finding representations that enable systematic optimization over complex design spaces. Differentiable architecture search relaxes discrete architectural choices into continuous variables (Liu et al., 2019), while molecular generative models enable property optimization in learned latent spaces (Jin et al., 2018). However, discrepancies between continuous formulations and their discrete realizations can undermine the improvements obtained during search (Chu et al., 2020). These longstanding optimization challenges persist in agentic workflows that bridge LLMs with external tools and other components (*i.e.*, developing principled and automated optimization), as discussed by Yuksekgonul et al. (2024).

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.

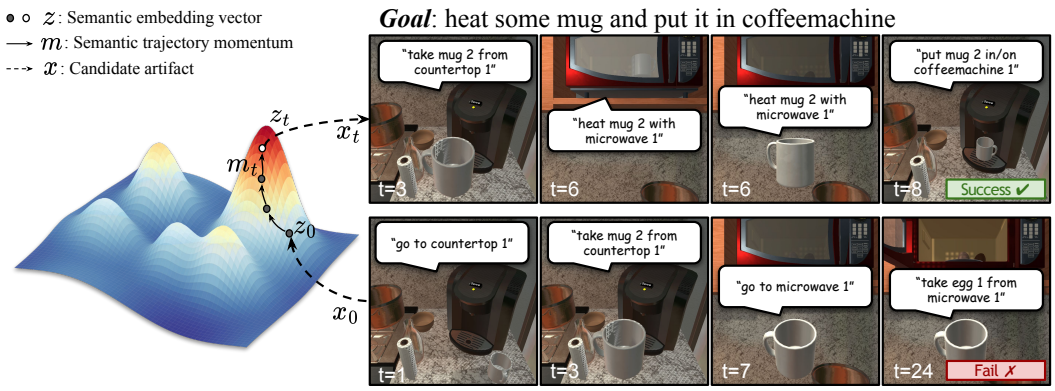


Figure 1: Semantic Relaxation for agentic workflows. By aligning descriptions and artifacts in a joint embedding space, Barycentric Semantic Optimization integrates candidate performance with trajectory momentum to guide description and artifact refinement. In this ALFWorld example, the initial parent x_0 (bottom) stalls on invalid actions, whereas the refined child x_t (top) successfully heats the mug and places it in the coffee machine. The schematic landscape illustrates the transition from the parent representation z_0 to the child representation z_t.

In this work, we carefully design **SemanticRelaxation**, an optimization framework that refines discrete generation through semantic updates within a joint description–artifact space, as illustrated in Fig. 1. Specifically, SemanticRelaxation consists of two core components. First, *Semantic Representation Learning* (SRL) establishes a joint embedding space that captures semantic relationships between descriptions and their corresponding artifacts. Then, SRL promotes cross-modal correlation and reduces feature redundancy, while a fuzzy topological regularizer preserves semantic neighborhood structure. This representation places discrete artifacts within a continuous semantic space, providing a consistent geometry for subsequent optimization. Second, *Barycentric Semantic Optimization* (BSO) derives explicit update directions from the geometry and task performance of evaluated candidates. BSO constructs a coordinated local semantic frame and computes a feedback-aligned barycentric direction toward favorable candidates. Moreover, it accumulates historical updates through trajectory momentum after parallel transport to maintain directional consistency across generations. The resulting semantic gradient guides semantic retrieval, through which refinement instructions are grounded in evaluated contextual trajectories and translated into artifact modifications. Taken together, SRL and BSO effectively project discrete artifacts into a joint semantic embedding space and exploit execution outcomes to explicitly direct iterative refinement through native generation interfaces, while preserving the task evaluation protocol.

Finally, SemanticRelaxation achieves state-of-the-art performance on the six agentic workflow benchmarks, outperforming the strongest competing baseline by up to 12.2 percentage points. Furthermore, code generation experiments show that SemanticRelaxation improves functional correctness while efficiently reducing overall token consumption by up to 29.2%. It is noteworthy that complementary quantitative and qualitative analyses on neural architecture search and molecular editing validate the joint semantic space and its contribution to artifact optimization.

Our main contributions are summarized as follows:

- We introduce Semantic Representation Learning to establish a joint description–artifact space through cross-modal correlation, redundancy reduction, and preservation of semantic neighborhoods, providing a geometric basis for semantic optimization.
- We propose Barycentric Semantic Optimization, which derive semantic update directions from evaluated candidates through barycentric aggregation and trajectory momentum, connecting continuous updates with discrete artifact generation.
- Experiments demonstrate leading performance on multiple agentic workflow benchmarks and assess the accuracy and token consumption of code generation. Additional studies on neural architecture search and molecular editing investigate representation quality and optimization behavior across distinct artifact structures.

2 METHOD

In recent multi-agent workflows, optimization loop is often guided by natural-language instructions, whereas performance is evaluated directly on the resulting artifacts (Li et al., 2026; Shang et al., 2025; Zhang et al., 2025a). However, these approaches do not inherently establish an explicit correlation between textual revisions and the resulting structural or functional changes, and consequently struggle to translate evaluation signals into reliable optimization guidance for effective agent refinement. To this end, we introduce SemanticRelaxation, a framework comprising two main modules. First, Semantic Representation Learning (Sec. 2.2) seamlessly aligns task-specific descriptions with corresponding candidates in a joint embedding space through cross-modal correlation and topology preservation. Barycentric Semantic Optimization (Sec. 2.3) then drives subsequent generation by effectively leveraging coordinated geometric relationships with trajectory momentum.

2.1 PROBLEM FORMULATION

We formulate semantic optimization in agentic workflows as an optimization problem over a discrete search space. Given task context c, let $x \in \mathcal{X}(c)$ denote a discrete artifact, $V(x; c)$ its validity indicator, and $U(x; c)$ its task evaluation score. We ultimately seek a valid artifact that maximizes Eq. (1):

$$x^\star \in \arg \max_{x \in \mathcal{X}(c):\, V(x;c)=1} U(x; c). \tag{1}$$

At iteration t, the current artifact x_t is associated with a natural-language description d_t, which summarizes the task objectives and the artifact's macro-level properties, whereas a refinement instruction r specifies how an artifact should be semantically modified. Then, each generated candidate $x_{t,i}$ is evaluated to obtain its performance $y_{t,i} = U(x_{t,i}; c)$ and execution history $f_{t,i}$. Together with the corresponding refinement instructions, these observations form the candidate set $\mathcal{B}_t = \{(r_{t,i}, x_{t,i}, y_{t,i}, f_{t,i})\}_{i=1}^{n_t}$, where all candidates share the task context and evaluation protocol. The execution history records the behavior of an individual candidate, whereas the optimization trajectory comprises successive refinement steps across iterations.

The central problem is to drive refinement instructions that translate observed performance differences into valid semantic improvements. Candidate evaluations establish relative performance, but do not directly determine which semantic modifications should be retained, revised, or extended. We therefore guide subsequent improvement through semantic optimization, computing semantic update directions from candidate changes within a joint description–artifact space and leveraging momentum accumulated along the optimization trajectory.

2.2 SEMANTIC REPRESENTATION LEARNING

Given pairs (d_i, x_i), frozen language and domain encoders E_d, E_x and trainable, modality-specific projection heads define as follows:

$$z_i^d = \nu(P_d(E_d(d_i))), \qquad z_i^x = \nu(P_x(E_x(x_i))) \in \mathbb{S}^{D-1}, \tag{2}$$

where $\nu(v) = v/\|v\|_2$ denotes ℓ_2 normalization for $v \neq 0$; domains without pretrained encoders use canonical structural features. Note that representations are learned separately for each domain. We promote cross-modal correlation between the projected representations while preserving semantic structure and reducing redundancy across feature dimensions through $\mathcal{L}_{\text{align}}$:

$$C = (\bar{Z}^d)^\top \bar{Z}^x / B, \quad \mathcal{L}_{\text{align}} = \frac{1}{D}\sum_a (C_{aa} - 1)^2 + \frac{\lambda_{\text{red}}}{D}\sum_{a \neq b} C_{ab}^2, \tag{3}$$

where C denotes the cross-correlation matrix between the projected views for a batch of B one-to-one pairs, with each feature dimension standardized across the batch. Here, the diagonal penalty promotes cross-modal agreement (*i.e.*, driving C_{aa} toward 1), while the off-diagonal penalty reduces redundancy between feature dimensions (*i.e.*, driving C_{ab} toward 0 for $a \neq b$) (Zbontar et al., 2021). To further shape the semantic geometry of aligned relationships, we present a topology regularizer:

$$q_{ij} = (1 + a\|z_i^x - z_j^x\|_2^{2b})^{-1} \text{ with } a, b > 0, \quad \mathcal{L}_{\text{top}} = -\tfrac{1}{2}\mathbb{E}_{\mathcal{E}^+}[p_{ij} \log q_{ij}] - \tfrac{1}{2}\mathbb{E}_{\mathcal{E}^-}[\log(1 - q_{ij})], \tag{4}$$

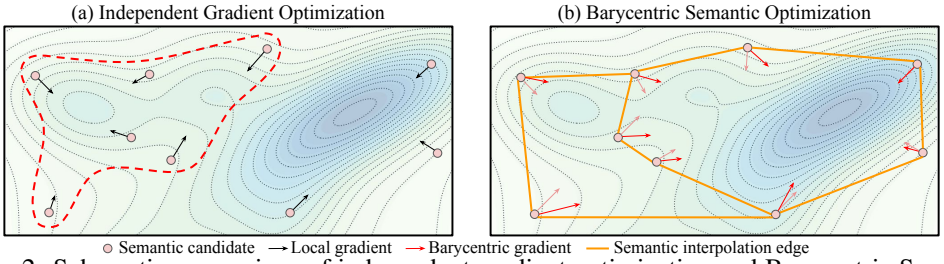


Figure 2: Schematic comparison of independent gradient optimization and Barycentric Semantic Optimization (BSO). (a) Independent local updates may produce inconsistent search directions. (b) BSO aggregates information across semantic candidates to derive barycentric update directions. Independent updates may lead to conflicting directions and stagnation near local optima. By contrast, BSO jointly coordinates candidate information to explore more promising regions.

where p_{ij} denotes a fixed neighborhood affinity computed from frozen description embeddings, and $\mathbb{E}$ represents uniform averaging over pairs $i \neq j$ with $p_{ij} > 0$ for $\mathcal{E}^+$ and $p_{ij} = 0$ for $\mathcal{E}^-$. Finally, we jointly optimize alignment and topology preservation through Eq. (5):

$$\mathcal{L}_{\mathrm{SRL}} = \mathcal{L}_{\mathrm{align}} + \lambda_{\mathrm{top}} \mathcal{L}_{\mathrm{top}}, \tag{5}$$

where λ_{top} is the coefficient of topology regularization. Especially, $\mathcal{L}_{\mathrm{SRL}}$ is optimized without supervision from task performance or improvement labels. The encoders and projection heads remain fixed throughout subsequent artifact optimization. This joint space advocates cross-modal alignment and local semantic structure, serving as the geometric basis for semantic optimization.

2.3 Barycentric Semantic Optimization

It is noteworthy that dependencies between semantic shifts and task-specific performance, suggested by evaluated candidates, provide sufficiently informative guidance for subsequent proposals, as detailed in Fig. 2. Let $z_t = z^x(x_t)$ and $z_{t,i} = z^x(x_{t,i})$ denote the embeddings of the current artifact and its i-th candidate, respectively. To account for anisotropy in the distribution of candidate embeddings, we adopt regularized whitening to the centered candidate embeddings, yielding $\widetilde{z}_{t,i}$, as provided in Appendix C. We then assign candidate weights according to task-specific evaluation signals and compute the offset of their weighted barycenter from the uniformly weighted center:

$$\begin{aligned} \alpha_{t,i} &= \frac{\exp\left((y_{t,i} - \bar{y}_t)/(\tau s_t)\right)}{\sum_{j=1}^{n_t} \exp\left((y_{t,j} - \bar{y}_t)/(\tau s_t)\right)}, \\ b_t &= \sum_{i=1}^{n_t} \left(\alpha_{t,i} - \frac{1}{n_t}\right) \widetilde{z}_{t,i}, \qquad z_{t+1}^{\mathrm{loc}} = \nu\left(z_t + \eta \tanh(\|b_t\|_2) \frac{b_t}{\|b_t\|_2}\right), \end{aligned} \tag{6}$$

where n_t is the number of evaluated candidates, $y_{t,i}$ is their task performance (*e.g.*, success rate, accuracy), and $\bar{y}_t = n_t^{-1} \sum_i y_{t,i}$. s_t normalizes performance variation, while $\tau > 0$ determines the concentration of candidate weights. z_{t+1}^{loc} denotes a local semantic target for the next iteration, where $0 < \eta < 1$ is the step size and $\nu(v) = v/\|v\|_2$ restores the embedding to the unit sphere. The $\tanh$ factor effectively bounds the step magnitude and suppresses updates when the aggregate direction is weak. Equal candidate performance yields equal-valued weights and $b_t = 0$, in which case we set $z_{t+1}^{\mathrm{loc}} = z_t$. To maintain directional information across iterations, we further accumulate BSO steps along the parent–child lineage of the current artifact. Let $\mathcal{H}_t = \{t - L_t + 1, \ldots, t\}$ denote the retained history indices, where $L_t \leq H$ and H is the history length. The retained local BSO steps are $\{(z_k, z_{k+1}^{\mathrm{loc}})\}_{k \in \mathcal{H}_t}$, including the current step $(z_t, z_{t+1}^{\mathrm{loc}})$. Since these steps originate at different positions on the sphere, we transport their directions to the tangent space at z_t before aggregation:

$$\begin{aligned} \xi_k &= \mathrm{Log}_{z_k}\left(z_{k+1}^{\mathrm{loc}}\right), \qquad \omega_{t,k} = \frac{\beta^{t-k}}{\sum_{\ell \in \mathcal{H}_t} \beta^{t-\ell}}, \\ m_t &= \sum_{k \in \mathcal{H}_t} \omega_{t,k} \, \mathrm{PT}_{z_k \to z_t}(\xi_k), \qquad \widehat{z}_{t+1} = \mathrm{Exp}_{z_t}(m_t). \end{aligned} \tag{7}$$

Here, Log_{z_k} expresses each local BSO step as a tangent vector ξ_k, and $\mathrm{PT}_{z_k \to z_t}$ transports it to the current tangent space. The coefficient $0 < \beta < 1$ determines the decay of historical contributions.

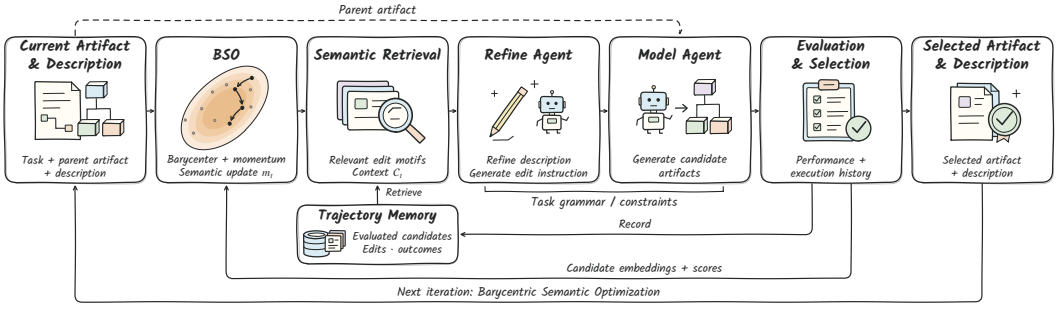


Figure 3: **Overview of SemanticRelaxation**. BSO integrates candidate performance and trajectory momentum in the semantic embedding space to guide retrieval. $\mathcal{R}_\theta$ uses retrieved edit motifs to refine the current description into a modification instruction, which $\mathcal{G}_\phi$ applies to generate candidate artifacts under task constraints. Evaluation updates the trajectory memory and candidate evidence for subsequent BSO updates, while the selected artifact becomes the parent for the next iteration.

We preserve the magnitude of m_t so that the aggregate update reflects directional alignment, with aligned directions contributing cumulatively and opposing directions partially or fully canceling one another. As a result, $\widehat{z}_{t+1} - z_t$ guides trajectory retrieval, with retrieved trajectories and candidate comparisons supporting subsequent instruction generation.

2.4 Detailed Design of SemanticRelaxation

SemanticRelaxation operates alongside existing multi-agent workflows as an explicit optimization module, effectively bridging semantic guidance and artifact generation through their native interfaces while preserving the evaluation protocol. SemanticRelaxation translates BSO updates into artifact modifications through semantic retrieval and two stages of generation, with resulting evaluations guiding subsequent updates to establish an iterative optimization loop, as illustrated in Fig. 3. Given current artifact embedding z_t and BSO target $\widehat{z}_{t+1}$, retrieval considers current artifact and direction $\widehat{z}_{t+1} - z_t$ to identify relevant edit motifs from the trajectory library. Each motif corresponds to an evaluated parent–instruction–artifact transition with recorded performance and execution history. Retrieved successful and unsuccessful transitions, together with evaluated candidate comparisons, constitute the context $\mathcal{C}_t$. In Eq. (8), Refine Agent $\mathcal{R}_\theta$ uses $\mathcal{C}_t$ to refine a current description into a modification instruction, and Model Agent $\mathcal{G}_\phi$ then generates corresponding artifact $x_{t+1,j}$:

$$r_{t+1,j} = \mathcal{R}_\theta(c, d_t, x_t, \mathcal{C}_t), \quad x_{t+1,j} = \mathcal{G}_\phi(x_t, r_{t+1,j}, \mathcal{S}_c), \tag{8}$$

where c denotes the task context, including its objectives and specification; x_t is the parent artifact; and d_t summarizes its task objectives and macro-level properties. The task grammar and validity constraints are specified by $\mathcal{S}_c$. For candidate index j, $r_{t+1,j}$ describes the intended modification, and $x_{t+1,j}$ is the generated artifact. Thus, BSO conditions generation through retrieved evidence and the resulting refinement instruction. Task evaluation yields performance $y_{t+1,j}$ and execution history $f_{t+1,j}$ for each candidate. These observations update the trajectory library and candidate evidence, preserving the association between each instruction and its realized outcome. Selected artifact becomes the parent for the next iteration, while BSO uses the updated evidence and retained trajectory momentum to compute the subsequent semantic target, remaining learned representation.

3 Experiments

3.1 Experimental setup

Benchmarks. We evaluate SemanticRelaxation across four domains: (1) **agent workflows**, including WebShop (Yao et al., 2022), ALFWorld (Shridhar et al., 2021), SciWorld (Wang et al., 2022), M3Tool (Wang et al., 2024b), Travel (Xie et al., 2024), and PDDL (Ma et al., 2024), covering web interaction, embodied execution, tool use, and planning; (2) **molecular editing**, involving increases and decreases in LogP, TPSA, and QED under the loose and strict settings considered in AgentDrug (Le et al., 2025); (3) **code generation**, including HumanEval (Chen et al., 2021a) and MBPP (Austin et al., 2021); and (4) **neural architecture search**, using NAS-Bench-201 on

Table 1: **Overall performance comparison** against hand-crafted agent and automated agent search methods. We evaluates on online shopping (WebShop), household task execution (ALFWorld), interactive science experiments (SciWorld), use of multiple tools (M3Tool), travel planning (Travel), and symbolic planning (PDDL). We report task score for WebShop, success rate for ALFWorld and M3Tool, progress rate for SciWorld and PDDL, and micro pass rate for Travel.

Method	Web	Embodied		Tool		Game
	WebShop	Alfworld	SciWorld	M3Tool	Travel	PDDL
CoT (Wei et al., 2022)	0.490	0.512	0.398	0.427	0.433	0.427
CoT-SC (Wang et al., 2023a)	0.488	0.545	0.412	0.451	0.410	0.410
ToT (Yao et al., 2023)	0.462	0.530	0.384	0.463	0.407	0.433
FoA (Klein et al., 2025)	0.509	0.587	0.427	0.488	0.474	0.472
TP (Yu et al., 2024)	0.343	0.373	0.195	0.402	0.387	0.440
SelfRefine (Madaan et al., 2023)	0.425	0.575	0.375	0.463	0.047	0.412
Dilu (Wen et al., 2024)	0.492	0.451	0.358	0.476	0.360	0.417
Voyager (Wang et al., 2024a)	0.423	0.336	0.389	0.317	0.517	0.337
DEPS (Wang et al., 2023b)	0.308	0.493	0.435	0.329	0.523	0.463
Stepback (Zheng et al., 2024)	0.459	0.470	0.314	0.488	0.033	0.403
ADAS (Hu et al., 2025)	0.436	0.567	0.463	0.500	0.453	0.509
MaAS (Zhang et al., 2025a)	0.485	0.612	0.437	0.537	0.403	0.564
AFlow (Zhang et al., 2025b)	0.497	0.619	0.452	0.524	0.497	0.528
AgentSquare (Shang et al., 2025)	0.520	0.701	0.475	0.561	0.553	0.577
AgentSwift (Li et al., 2026)	<u>0.562</u>	**0.806**	<u>0.509</u>	<u>0.634</u>	<u>0.573</u>	**0.614**
SemanticRelaxation	**0.640**	<u>0.784</u>	**0.579**	**0.756**	**0.660**	<u>0.580</u>

CIFAR-10, CIFAR-100, and ImageNet16-120 (Dong & Yang, 2020). We report the respective task metrics: benchmark-specific execution performance (*i.e.*, success rate, progress rate, or pass rate) for agent workflows, validity and property satisfaction for molecular editing, functional correctness for code generation, and validation and test accuracy for neural architecture search. Note that we adopt `GPT-4o-mini` for agent workflows and code generation, and `Llama-3.1-8B-Instruct` for molecular editing and neural architecture search. More implementation details in Appendix A.

Baselines. We group the baselines by their reasoning, coordination, and optimization mechanisms. (i) **Single-agent reasoning and search** baselines include CoT (Wei et al., 2022), ComplexCoT (Fu et al., 2023), SC/CoT-SC (Wang et al., 2023a), ToT (Yao et al., 2023), TP (Yu et al., 2024), SelfRefine (Madaan et al., 2023), Stepback (Zheng et al., 2024), MultiPersona (Wang et al., 2024c), GENIUS (Zheng et al., 2023), and LLMatic (Nasir et al., 2024). MultiPersona simulates collaboration among personas within a single LLM, while GENIUS and LLMatic use an LLM for iterative or evolutionary search. (ii) **Hand-crafted agent and search workflows** include Dilu (Wen et al., 2024), Voyager (Wang et al., 2024a), DEPS (Wang et al., 2023b), ChatDrug (Liu et al., 2024), AgentDrug (Le et al., 2025), and RZ-NAS (Ji et al., 2025). (iii) **Hand-crafted multi-agent** workflows comprise FoA (Klein et al., 2025), LLM-Debate (Du et al., 2024), DyLAN (Liu et al., 2023), AgentVerse (Chen et al., 2024b), and MacNet (Qian et al., 2025), while LLM-Blender (Jiang et al., 2023) represents output ensembling. (iv) **Automated agent and workflow design** methods include AutoAgents (Chen et al., 2024a), GPTSwarm (Zhuge et al., 2024), ADAS (Hu et al., 2025), AgentSquare (Shang et al., 2025), AgentSwift (Li et al., 2026), AFlow (Zhang et al., 2025b), and MaAS (Zhang et al., 2025a), which automate agent construction or workflow optimization.

3.2 Overall Performance

We aim to facilitate continuous optimization in complex tasks by aligning task-specific descriptions with their corresponding artifacts. To evaluate the effectiveness of SemanticRelaxation, we first conduct quantitative comparisons with existing methods on agentic workflow and code generation benchmarks. For fair comparison, we use the same GPT-4o-mini backbone as the baselines.

Agentic Workflows. We evaluate SemanticRelaxation on complex tasks involving web interaction, embodied execution, tool use, and planning to assess the effectiveness of continuous optimization through semantically aligned representations. As reported in Tab. 1, SemanticRelaxation achieves the highest performance among the compared methods on four of the six benchmarks: WebShop, SciWorld, M3Tool, and Travel. In particular, the success rate on M3Tool increases from 63.4%

Table 2: **Evaluation on HumanEval and MBPP** with single-agent methods, hand-crafted multi-agent systems and automated agentic workflows.

Method	HumanEval	MBPP
Vanilla	87.08	71.83
CoT (Wei et al., 2022)	88.13	71.83
ComplexCoT (Fu et al., 2023)	87.49	72.36
SC (CoT×5) (Wang et al., 2023a)	88.60	73.60
MultiPersona (Wang et al., 2024c)	88.32	73.19
LLM-Debate (Du et al., 2024)	88.68	70.29
LLM-Blender (Jiang et al., 2023)	88.80	77.05
DyLAN (Liu et al., 2023)	90.42	77.30
AgentVerse (Chen et al., 2024b)	89.29	74.28
AutoAgents (Chen et al., 2024a)	87.64	71.95
GPTSwarm (Zhuge et al., 2024)	89.32	77.43
ADAS (Hu et al., 2025)	84.19	68.13
AgentSquare (Shang et al., 2025)	89.08	78.46
AFlow (Zhang et al., 2025b)	90.93	81.67
MaAS (Zhang et al., 2025a)	92.85	82.17
SemanticRelaxation	**93.89**	**87.68**

Table 3: **Computational overhead on HumanEval** under a unified training/inference protocol (RTX A6000 GPU). Tokens and time are reported in millions (M) and hours (h). API costs use GPT-4o-mini rates. Red percentages denote reductions relative to MaAS.

Category	Component	MaAS	SemanticRelaxation
Training	Prompt	7.83M	**6.10M** (-22.07%)
	Completion	2.10M	**1.46M** (-30.32%)
	Total	9.92M	**7.56M** (-23.82%)
Inference	Prompt	2.57M	**1.41M** (-44.86%)
	Completion	0.76M	**0.41M** (-46.41%)
	Total	3.32M	**1.82M** (-45.22%)
Cost (USD)	Training	$2.43	**$1.79** (-26.34%)
	Inference	$0.84	**$0.46** (-45.70%)
	Total	$3.27	**$2.25** (-31.31%)
Wall-clock	Training	6.67h	**4.65h** (-30.18%)
	Inference	2.20h	**1.43h** (-34.86%)
	Total	8.87h	**6.09h** (-31.34%)

to 75.6%, outperforming AgentSwift by 12.2 percentage points. Although macro-level semantic directions may not fully specify the concrete action sequences required for embodied execution and symbolic planning, potentially limiting performance on ALFWorld and PDDL, SemanticRelaxation showcases consistent improvements across all six benchmarks, supporting the effectiveness of SemanticRelaxation across multiple task settings. We demonstrate that semantically aligned continuous optimization represents a promising approach to improving automated agentic workflows.

Code Generation. We also present the results on HumanEval and MBPP, where SemanticRelaxation achieves the highest reported scores of 93.89% and 87.68%, respectively, as shown in Tab. 2. Compared with MaaS, the strongest baseline on both benchmarks, our method improves performance by 1.04 percentage points on HumanEval and 5.51 percentage points on MBPP. Especially, the gains over Vanilla are 6.81 and 15.85 percentage points, respectively. We finally validate that improvements over both the vanilla baseline and existing automated workflows support the applicability of SemanticRelaxation to code generation, with a larger advantage on MBPP.

We further evaluate the computational efficiency of SemanticRelaxation on HumanEval by comparing token consumption and wall-clock time with MaaS under a unified experimental protocol. By efficiently guiding automated workflows, SemanticRelaxation reduces unnecessary LLM calls and lowers computational overhead during both training and inference. As reported in Tab. 3, inference token consumption and wall-clock time decrease by 45.22% and 34.86%, respectively, demonstrating improved execution efficiency. More experiments are provided in Appendix Tab. 8.

3.3 Ablation Study

To better understand the role of each module in SRL, we present ablation studies, as shown in Tab. 4. To this end, we construct a held-out set of 42 ALFWorld workflows and identify 38 eligible workflow pairs that differ in exactly one module (*e.g.*, planning, reasoning, or memory). We consider four semantic evaluation scenarios: (1) *Original* retrieves the corresponding artifact from its original description; (2) *Rephrased* evaluates retrieval after meaning-preserving changes in clause order and terminology; (3) *Meaning changed* examines whether a one-module modification redirects retrieval to the corresponding workflow; and (4) *Wrong-match rejection* removes all valid matches and evaluates whether incorrect artifacts are rejected. All samples are selected according to workflow specifications without using task performance.

Table 4: Ablation study in SRL under semantic perturbation. Values are the condition-specific success rate (%) on the fixed test queries; n denotes the number of eligible queries for each check.

Semantic Perturbation	n	$C_{aa} \to 1$	w/ $C_{ab} \to 0$	w/ $\mathcal{L}_{top}$
Original	42	59.52	61.90	66.67
Rephrased	42	14.29	16.67	35.71
Meaning changed	38	28.95	44.74	50.00
Wrong-match rejection	42	9.52	9.52	14.29

Table 5: **Comparison on NAS-Bench-201** with representative conventional methods and hand-crafted agent methods. Validation and test accuracies (%) are reported as mean ± standard deviation over four independent search runs. *Optimal* denotes the highest mean accuracy for each dataset and split within the search space. ∗ and † represent GPT-4o and Llama-3.1-8B, respectively.

Method	CIFAR-10 (↑)		CIFAR-100 (↑)		ImageNet16-120 (↑)	
	Val.	Test	Val.	Test	Val.	Test
Optimal	91.61	94.37	73.49	73.51	46.77	47.31
RS (Bergstra & Bengio, 2012)	90.93 ± 0.36	93.70 ± 0.36	70.93 ± 1.09	71.04 ± 1.07	44.45 ± 1.10	44.57 ± 1.25
REA (Real et al., 2018)	91.19 ± 0.31	93.92 ± 0.30	71.81 ± 1.12	71.84 ± 0.99	45.15 ± 0.89	45.54 ± 1.03
REINFORCE (Williams, 1992)	91.09 ± 0.37	93.85 ± 0.37	71.61 ± 1.12	71.71 ± 1.09	45.05 ± 1.02	45.24 ± 1.18
BOHB (Falkner et al., 2018)	90.82 ± 0.53	93.61 ± 0.52	70.74 ± 1.29	70.85 ± 1.28	44.26 ± 1.36	44.42 ± 1.49
DrNAS (Chen et al., 2021b)	$\mathbf{91.55 \pm 0.00}$	$\mathbf{94.36 \pm 0.00}$	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	46.37 ± 0.00	$\mathbf{46.34 \pm 0.00}$
β-DARTS (Ye et al., 2022)	$\mathbf{91.55 \pm 0.00}$	$\mathbf{94.36 \pm 0.00}$	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	46.37 ± 0.00	$\mathbf{46.34 \pm 0.00}$
Λ-DARTS (Movahedi et al., 2023)	$\mathbf{91.55 \pm 0.00}$	$\mathbf{94.36 \pm 0.00}$	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	46.37 ± 0.00	$\mathbf{46.34 \pm 0.00}$
GENIUS* (Zheng et al., 2023)	91.07 ± 0.20	93.79 ± 0.09	70.96 ± 0.33	70.91 ± 0.72	45.29 ± 0.81	44.96 ± 1.02
LLMatic* (Nasir et al., 2024)	–	94.26 ± 0.13	–	71.62 ± 1.73	–	45.87 ± 0.96
RZ-NAS* (Ji et al., 2025)	91.45 ± 0.10	94.24 ± 0.12	73.35 ± 0.14	73.30 ± 0.21	$\mathbf{46.53 \pm 0.24}$	46.24 ± 0.23
SemanticRelaxation†	$\mathbf{91.55 \pm 0.00}$	$\mathbf{94.36 \pm 0.00}$	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	46.37 ± 0.00	$\mathbf{46.34 \pm 0.00}$

Enforcing $C_{aa} \to 1$ alone establishes description–artifact alignment, while evaluation under semantic perturbations quantitatively reveals the complementary benefits of redundancy reduction and topology preservation. In particular, incorporating $\mathcal{L}_{top}$ improves success on rephrased queries from 16.67% to 35.71% and wrong-match rejection from 9.52% to 14.29%, suggesting that preserving semantic relationships helps organize the joint embedding space under variations in wording and match availability. Reducing representational redundancy through $C_{ab} \to 0$ improves success on meaning-changed queries from 28.95% to 44.74%, suggesting more distinct positioning of semantic frames associated with individual module changes. SRL therefore benefits from combining cross-modal alignment with redundancy reduction and topology preservation to improve robustness to meaning-preserving perturbations while retaining sensitivity to meaningful modifications.

3.4 Optimization in the Joint Semantic Space

We showcase that the learned semantic space practically provides meaningful directions for improving discrete artifacts. Experiments on NAS-Bench-201 and AgentDrug assess both the effectiveness of semantic updates and the relationship between embedding geometry and task objectives.

Neural Architecture Search. SemanticRelaxation achieves test accuracies of 94.36%, 73.51%, and 46.34% on CIFAR-10, CIFAR-100, and ImageNet16-120, respectively, matching the strongest search methods compared in Tab. 5. In particular, SemanticRelaxation reaches the reported search-space optimum on CIFAR-100 and falls only 0.01 percentage points below the optimum on CIFAR-10. Competitive performance against specialized NAS methods demonstrates that the proposed domain-agnostic framework can effectively guide architecture generation through semantic optimization.

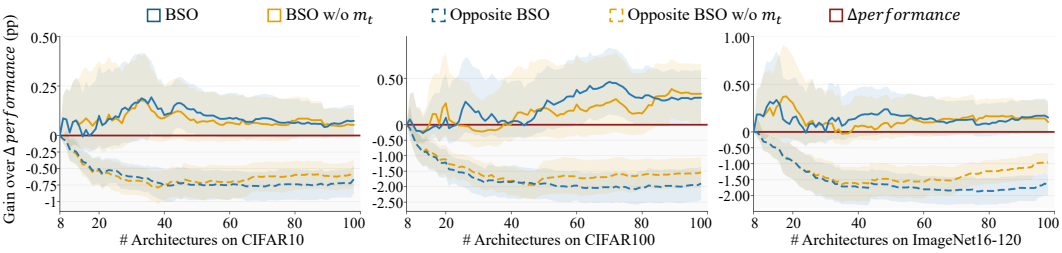
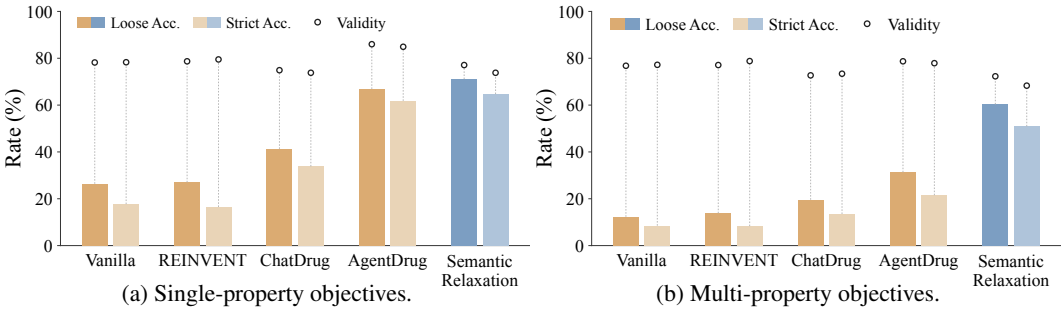


Figure 4: **Performance gains over Δ *performance*.** Note that Δ *performance* directly guides search in the semantic representation space using only performance differences.

Effect of Semantic Update Directions. We compare BSO with updates in the opposite semantic direction to examine whether performance gains arise from meaningful directional alignment (Fig. 4). BSO improves over the Δ *performance* baseline across all three datasets (up to 0.33 percentage points), whereas reversing the direction incurs losses of up to 2.19 percentage points. It is noteworthy that opposite-direction losses significantly support that BSO gains arise from aligning semantic updates with task objectives rather than incidental improvements from candidate generation.



(a) Single-property objectives. (b) Multi-property objectives.

Figure 5: **Molecular editing performance on AgentDrug** using Llama-3.1-8B-Instruct. Bars show property-optimization accuracy under loose and strict settings, while circle markers indicate molecular validity. Faint connectors visualize the gap between accuracy and validity. Results are averaged over the corresponding editing objectives. Detailed results in Tabs. 6 and 7

Molecular Editing. SemanticRelaxation improves average single-objective accuracy over AgentDrug from 66.9% to 71.1% under loose criteria and from 61.8% to 64.7% under strict criteria (Figure 5). Larger improvements emerge when multiple properties must be satisfied simultaneously: multi-objective accuracy increases from 31.3% to 60.3% and from 21.5% to 50.9%, respectively. We show that these gains under joint property requirements demonstrate effective refinement of molecular artifacts toward specified objectives. Detailed results are provided in Appendix Tabs. 6 and 7

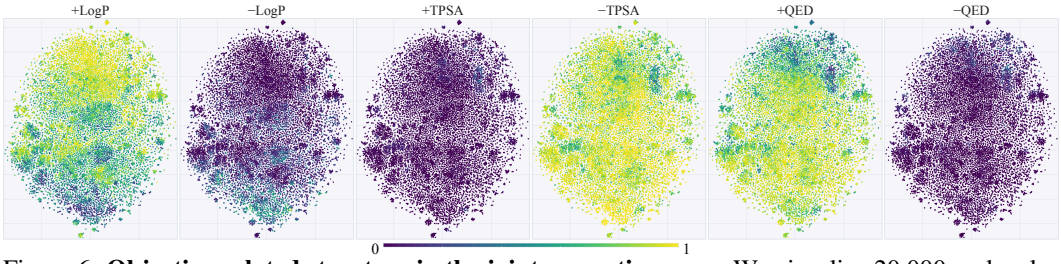


Figure 6: **Objective-related structure in the joint semantic space.**. We visualize 20,000 molecules across six property objectives to examine whether semantically similar samples cluster together and how their neighborhoods relate to optimization performance. Colors represent local accuracy, measured as the fraction of the 50 nearest neighboring molecules satisfying each objective relative to a fixed reference molecule.

Objective Structure in the Molecular Retrieval Space. We visualize 20,000 molecules using t-SNE on retrieval distances, defined as one minus the Tanimoto similarity between radius-2 Morgan sparse-count fingerprints, as shown in Fig. 6. Colors indicate the fraction of each molecule's 50 nearest neighbors under the original retrieval distance that satisfy the target objective relative to a fixed reference molecule; objective outcomes affect only colors, not coordinates. We note that high-satisfaction neighborhoods vary with the target property and optimization direction, suggesting that retrieval similarity captures objective-relevant molecular relationships.

4 Conclusion

We presented SemanticRelaxation, a domain-agnostic framework for optimizing discrete artifacts through a joint description–artifact space. Semantic Representation Learning aligns descriptions with artifacts while reducing redundancy and preserving semantic neighborhoods. Barycentric Semantic Optimization combines candidate performance with trajectory momentum to derive semantic update directions for iterative refinement. Experiments demonstrate state-of-the-art performance on four agentic workflow benchmarks and improved code generation accuracy with reduced token consumption and wall-clock time. Complementary studies on neural architecture search and molecular editing, together with opposite-direction experiments, support the practical value of semantic optimization for discrete generation.

AI USE STATEMENT

LLMs are components of the proposed framework and experimental systems, as described in Secs. 2 and 3 and Appendix A. For manuscript preparation, we used AI tools for grammar and style polishing and figure preparation. We did not use AI tools to develop the research ideas, design the experiments, or interpret the results. The authors take full responsibility for all content.

ETHICS STATEMENT

This work evaluates automated refinement of agentic workflows, code, molecules, and neural architectures using existing benchmarks. Benchmark performance does not establish the safety of generated artifacts for real-world use. In particular, molecular property improvements do not establish biological safety, and successful code or agent execution does not guarantee secure or reliable behavior beyond the evaluated tasks. Downstream applications require domain-specific validation and safeguards against misuse.

REPRODUCIBILITY STATEMENT

The SRL, BSO, and refinement pipeline descriptions are provided in Secs. 2.2 to 2.4. Sec. 3.1 and Appendix A document the benchmarks, evaluation metrics, and domain-specific experimental settings. Fig. 7 provides pseudocode for the framework. Figs. 8 and 9 provide the ALFWorld prompts for the Refine Agent and Model Agent.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

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

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje Karlsson, Jie Fu, and Yemin Shi. Autoagents: A framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 22–30, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024b.

Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=9FWas6YbmB3.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations*, volume 2024, pp. 8746–8825, 2024c.

Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European conference on computer vision*, pp. 465–480. Springer, 2020.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2024. URL https://openreview.net/forum?id=QAwaaLJNCk.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446, 2018.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=9ZxnPZGmPU.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yf1icZHC-l9.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. Rz-nas: Enhancing llm-guided neural architecture search via reflective zero-cost strategy. In *ICML*, 2025.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Lars Henning Klein, Nearchos Potamitis, Roland Aydin, Robert West, Caglar Gulcehre, and Akhil Arora. Fleet of agents: Coordinated problem solving with large language models. In *The Exploration in AI Today Workshop at ICML 2025*, 2025. URL https://openreview.net/forum?id=wjEkZIZdQT.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pp. 1945–1954. PMLR, 2017.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

Khiem Le, Ting Hua, and Nitesh V. Chawla. Agentdrug: Utilizing large language models in an agentic workflow for zero-shot molecular editing. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24448–24458, 2025.

Yu Li, Lehui Li, Zhihao Wu, Qingmin Liao, Jianye Hao, Kun Shao, and Fengli Xu. Agentswift: Efficient llm agent design via value-guided hierarchical search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31843–31851, 2026.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. Conversational drug editing using retrieval and domain feedback. In *International Conference on Learning Representations*, volume 2024, pp. 48325–48361, 2024.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. *arXiv preprint arXiv:2310.02170*, 2023.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Sajad Movahedi, Melika Adabinejad, Ayyoob Imani, Arezou Keshavarz, Mostafa Dehghani, Azadeh Shakery, and Babak N Araabi. λ-DARTS: Mitigating performance collapse by harmonizing operation selection among cells. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=oztkQizr3kk.

Muhammad Umair Nasir, Sam Earle, Julian Togelius, Steven James, and Christopher W. Cleghorn. Llmatic: Neural architecture search via large language models and quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7957–7968, 2023.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. 2018. URL https://arxiv.org/pdf/1802.01548.pdf.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. In *International Conference on Learning Representations*, volume 2025, pp. 3841–3865, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024b. URL https://openreview.net/forum?id=8oJyuXfrPv.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 257–279, 2024c.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. *Advances in Neural Information Processing Systems*, 36:34153–34189, 2023b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language models. In *International Conference on Learning Representations*, volume 2024, pp. 34503–34522, 2024.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=kirH8xw6YA.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. β-darts: Beta-decay regularization for differentiable architecture search. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10864–10873, 2022. doi: 10.1109/CVPR52688.2022.01060.

Junchi Yu, Ran He, and Rex Ying. Thought propagation: An analogical approach to complex reasoning with large language models. In *International Conference on Learning Representations*, volume 2024, pp. 10989–11015, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pp. 12310–12320. PMLR, 2021.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=imcyVlzpXh.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025b.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 20279–20316, 2024.

Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. Can gpt-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970*, 2023.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=uTC9AFXIhg.

A Implementation Details

A.1 Agentic Workflow Benchmarks

We evaluate WebShop for online shopping, ALFWorld for household tasks, SciWorld for scientific experiments, M3Tool for tool use, TravelPlanner for itinerary planning, and PDDL for symbolic planning. Recorded evaluations use 50 or 500 WebShop instances, 50 or 134 ALFWorld instances, 26 SciWorld instances, 50 or 82 M3Tool instances, 30 TravelPlanner instances, and 60 PDDL instances. We report mean reward for WebShop, success rate for ALFWorld and M3Tool, progress for SciWorld and PDDL, and the released commonsense micro pass rate for TravelPlanner. GPT-4o-mini proposals use temperatures 0.8 for evolution and 0.1 for recombination, top-$p = 1$, and a 4,096-token output limit.

A.2 Molecular Editing

We follow AgentDrug (Le et al., 2025) on six single-property and eight dual-property objectives involving LogP, TPSA, and QED, with 500 molecules per objective and difficulty. Loose conditions require improvement in the requested direction; strict conditions require changes exceeding 0.5 LogP, $10\ \text{Å}^2$ TPSA, or 0.1 QED. Validity and accuracy are computed over all inputs, and accuracy requires satisfying every requested objective without an additional similarity threshold. Llama-3.1-8B-Instruct uses temperature 0, top-$p = 1$, and a 256-token output limit. Each input receives one initial generation and up to three refinements, supported by a 10,000-molecule retrieval database.

A.3 HumanEval and MBPP

We adopt 33/131 training/evaluation problems for HumanEval and 86/341 for MBPP. GPT-4o-mini uses execution temperature 1.0, textual-update temperature 0.8, top-$p = 1$, and a 4,096-token output limit. Our iterative configuration performs three training passes and selects a checkpoint using training-split assessments.

A.4 NAS-Bench-201

NAS-Bench-201 (Dong & Yang, 2020) contains $5^6 = 15{,}625$ cell architectures. We optimize tabulated 200-epoch validation accuracy on CIFAR-10, CIFAR-100, and ImageNet16-120, reporting test accuracy only after architecture selection. Llama-3.1-8B-Instruct uses temperature 0.8 and a 3,000-token output limit. Each search starts from 20 randomly sampled architectures and permits up to 30 rounds and 200 validation queries, stopping after five rounds without improvement. A round proposes eight descriptions and targets six evaluated candidates.

A.5 Implementation setup

During semantic representation learning, we freeze the description and artifact encoders and train their projectors using Barlow Twins alignment and topology regularization. All representation modules remain fixed during subsequent optimization, as shown in Fig. 7. At each iteration, BSO combines performance-conditioned updates with lineage momentum to retrieve relevant trajectories. Refine agent (Fig. 8) uses task evidence and retrieved transitions to produce refinement instructions, which model agent (Fig. 9) applies to generate complete candidate artifacts. Both agents return structured JSON outputs following task-specific constraints. Evaluation results then update the candidate pool and trajectory memory.

B Related Work

B.1 Automated Agentic Workflows.

LLM self-improvement has been explored through iterative feedback, reflection, textual gradients, and evolutionary or history-based optimization, where natural-language critiques or evaluated candidates guide subsequent generations (Madaan et al., 2023; Shinn et al., 2023; Pryzant et al., 2023; Yang et al.,

2024; Fernando et al., 2024; Yuksekgonul et al., 2025). More broadly, recent work expands the optimization target from individual prompts and outputs to complete agentic systems: GPTSwarm (Zhuge et al., 2024) optimizes agent graphs, ADAS (Hu et al., 2025) searches executable agent programs, AFlow (Zhang et al., 2025b) explores code-represented workflows with MCTS, and AgentSquare (Shang et al., 2025) evolves and recombines modular agent components. MaAS (Zhang et al., 2025a) instead optimizes a probabilistic continuous agentic supernet from which query-dependent architectures are sampled, while AgentSwift (Li et al., 2026) combines hierarchical agent design with learned value models and uncertainty-guided search. Despite increasingly sophisticated search mechanisms, these methods maintain search state through textual feedback, candidate populations, search trees, learned surrogates, or architecture distributions; they do not explicitly maintain a persistent numerical direction in a shared semantic space, estimated from realized candidate changes and accumulated across generations.

B.2 Structured Discrete Optimization and Continuous Relaxation.

LLMs have increasingly been used to refine structured artifacts under external feedback: execution signals guide program synthesis and self-debugging (Le et al., 2022; Chen et al., 2024c), retrieval and domain feedback support molecular editing (Liu et al., 2024), and linguistic reflection assists neural architecture search (Ji et al., 2025). A complementary line of work introduces continuous optimization structure into discrete spaces. Differentiable NAS relaxes architectural choices into continuous parameters or distributions (Liu et al., 2019; Xie et al., 2019; Chen et al., 2021b), while latent-variable models encode discrete molecular or symbolic structures into continuous representations for optimization and subsequent decoding (Kusner et al., 2017; Jin et al., 2018). Semantic Relaxation instead leaves the native discrete generator, validity constraints, and external evaluator intact, using continuous representations only to estimate and accumulate semantic directions from realized and evaluated candidate transitions.

C Regularized Whitening

Candidate embeddings may exhibit different degrees of dispersion across directions, causing barycentric aggregation to depend on their geometric scale. We apply regularized whitening to account for this variation before computing the semantic update. Given n_t evaluated candidate embeddings $\{z_{t,i}\}_{i=1}^{n_t} \subset \mathbb{R}^D$, we define their mean and empirical covariance as

$$\mu_t = \frac{1}{n_t}\sum_{i=1}^{n_t} z_{t,i}, \quad S_t = \frac{1}{n_t}\sum_{i=1}^{n_t}(z_{t,i}-\mu_t)(z_{t,i}-\mu_t)^\top. \tag{9}$$

Since $\mathrm{rank}(S_t) \le \min(D, n_t - 1)$, the covariance can be singular when the candidate set is small. We therefore restrict the transformation to the subspace spanned by the centered embeddings and regularize its scaling:

$$\widehat{S}_t = (1-\rho_t)S_t + \rho_t \frac{\mathrm{tr}(S_t)}{D} I_D, \quad W_t = V_t\left(V_t^\top \widehat{S}_t V_t + \epsilon_t I_{r_t}\right)^{-1/2} V_t^\top, \quad \widetilde{z}_{t,i} = W_t(z_{t,i}-\mu_t), \tag{10}$$

where $r_t = \mathrm{rank}(S_t) > 0$, $V_t \in \mathbb{R}^{D\times r_t}$ contains an orthonormal basis for the range of S_t, $\rho_t \in [0,1]$ is the covariance shrinkage coefficient, and $\epsilon_t > 0$ is the regularization parameter. The inverse square root is taken over the symmetric positive definite matrix within this subspace.

The transformation rescales the observed directions without introducing components outside the candidate span. Shrinkage moderates differences between covariance eigenvalues, while ϵ_t limits amplification along directions with small variance. Consequently, the transformed covariance is not generally the identity. Centering is preserved, $\frac{1}{n_t}\sum_i \widetilde{z}_{t,i} = 0$, so the uniform candidate distribution provides a zero baseline for barycentric aggregation. When $r_t = 0$, we set $W_t = 0$ and $\widetilde{z}_{t,i} = 0$. This transformation is computed for the current candidate set; the learned encoders and projection heads remain unchanged.

semantic_relaxation.py — PyTorch-style pseudocode

```python
import torch
import torch.nn.functional as F

# 1. Semantic Representation Learning
for desc, artifact in paired_data:
    with torch.no_grad():                    # frozen encoders
        hd, hx = Ed(desc), Ex(artifact)

    zd = F.normalize(Pd(hd), dim=-1)
    zx = F.normalize(Px(hx), dim=-1)
    loss = barlow_twins(zd, zx) + lambda_top * topology_loss(zx, hd)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

freeze(Ed, Ex, Pd, Px)

# 2. Barycentric Semantic Optimization
with torch.no_grad():
    for t in range(iterations):
        z = encode_artifact(x)
        Z, y = encode_artifact(B.artifacts), B.performance
        Z_white = regularized_whiten(Z - Z.mean(dim=0))

        # Performance-conditioned barycentric direction
        weights = torch.softmax((y - y.mean()) / (tau * scale(B)), dim=0)
        b = (weights - 1 / len(y)) @ Z_white
        strength = b.norm()
        z_local = F.normalize(
            z + eta * strength.tanh() * b / strength.clamp_min(1e-8),
            dim=-1,
        )

        # Transport and aggregate updates along the current lineage
        steps = (memory.lineage(x) + [(z, log_map(z, z_local))])[-H:]
        w = beta ** torch.arange(len(steps) - 1, -1, -1, device=z.device)
        directions = torch.stack([transport(p, z, v) for p, v in steps])
        m = ((w / w.sum())[:, None] * directions).sum(dim=0)

        # Retrieval → instruction → artifact → evaluation
        C = retrieve(memory, z, direction=exp_map(z, m) - z)
        r = refine_agent(c, d, x, C, S)
        X = model_agent(x, r, S)
        y, f = evaluate(X, c)

        memory.record(x, r, X, y, f, bso_step=steps[-1])
        B = update_candidates(B, X, r, y, f)
        x, d = select_candidate(B)

return x
```

SRL + BSO · Abstract helper functions — Python | UTF-8

Figure 7: PyTorch-style pseudocode for Semantic Relaxation. SRL learns a joint description–artifact representation through Barlow Twins alignment and topology regularization. With this representation fixed, BSO combines barycentric updates and transported trajectory momentum to guide semantic retrieval and two-stage artifact generation. Task evaluation updates the candidate evidence and trajectory memory for subsequent iterations.

Refine Prompt

TRANSPORT
Return one JSON object whose only top-level key is `descriptions`; its value is an array of JSON objects following the task-specific output schema in S_C. Emit each typed object directly: do not add the internal description prefix and do not encode an inner object as a JSON string. The transport canonicalizes each accepted object once into the prefixed text used by representation learning and artifact refinement. Return no Markdown or extra prose. Emit up to requested_description_count distinct items. Valid items survive retries; never pad with an invalid or duplicate item.

TASK
Role: You are an embodied household-agent policy expert for ALFWorld TextWorld. Improve one reusable observation-conditioned policy across all task families using aggregate valid_seen evidence and retrieved policy trajectories, without writing an episode walkthrough.

Goal: Improve aggregate valid_seen task completion using only policy-visible observations.
Artifact boundary: The artifact is a reusable text policy across task families, not an episode action sequence.

TASK SPECIFICATION
{S_C}

S_C provides the task-specific legal search space, grammar, constraints, rules, generation procedure, output fields and formats, and valid and invalid examples. Follow S_C when constructing each refinement instruction. Examples illustrate the task requirements and output structure; they are not evidence of measured improvement.

RUNTIME INPUTS
{TASK_EVIDENCE}, {CURRENT_DESCRIPTION},
{RETRIEVED_TRAJECTORY_EDITS}, {REQUESTED_DESCRIPTION_COUNT}

BSO OPTIMIZATION
1. task_evidence is authoritative for the live artifact and currently valid operands. S_C defines the task-specific rules and constraints.
2. retrieved_trajectory_edits contains transitions already selected for the current BSO direction, including lineage momentum when available. Compare each parent description, child description, realized edit, and validator feedback; when prose and execution differ, trust the realized edit.
3. Use parent_relative_delta only as observed support or counterevidence for the realized edit; never treat it as an edit label or a predicted score.
4. Infer the transferable relationship behind a useful transition and rebind it to the live parent. Never copy stale artifact values or historical identifiers.
5. Repeated compatible transitions indicate a direction sustained across the lineage. Treat isolated or conflicting transitions as controlled alternatives rather than combining them indiscriminately.
6. When no trajectory is retrieved, form distinct, local, task-grounded hypotheses from the live state instead of inventing optimization evidence.
7. If task_evidence.accepted_exact_description_identity_exclusions exists, avoid exact listed identities only.

FINAL CHECK
Use one concrete parent-relative change, legal live operands, and the output schema specified in S_C. Preserve unrelated valid structure. Do not return current_description unchanged. Do not expose optimizer labels, momentum weights, similarity values or latent coordinates, and do not invent performance.

Figure 8: Refine Agent $\mathcal{R}_\theta$ prompt.

D Additional Experiments

We provide detailed molecular editing results under single- and multi-objective settings and compare computational overhead on MBPP and HumanEval.

Model Prompt

TRANSPORT
Return one JSON object following the output schema in S_C. For this policy-generation interface, return: {"policies":[{"policy":"complete global policy"}]}. Return exactly one complete policy in this response. The runtime repeats this call to fill the requested pool. Return no Markdown or surrounding prose.

TASK
Role: You are an ALFWorld embodied-policy expert. Apply the supplied semantic policy correction to the current parent policy.
Goal: Generate a complete reusable global policy that implements the intended modification.
Artifact boundary: The output is the complete natural-language
policy, not another refinement instruction or an episode action sequence.

TASK SPECIFICATION
{S_C}

S_C provides the task-specific policy components, command grammar, constraints, rules, permitted information, generation procedure, output fields and formats, and valid and invalid examples. Follow S_C when interpreting the correction and constructing the policy. Examples illustrate the task requirements and output structure; they are not evidence of measured improvement.

RUNTIME INPUTS
{PARENT_GLOBAL_POLICY}, {ARTIFACT_REFINEMENT_DIRECTIVE}, {CANDIDATE_COUNT}, {EXCLUDED_POLICIES}, {GENERATION_ROUND}, {SEMANTIC_UPDATE_INDEX}, {ATTEMPT}

ARTIFACT GENERATION
1. Read the current parent policy and the requested semantic correction. artifact_refinement_directive.refined_description and its deterministic semantic_fields represent one correction. Interpret the fields according to S_C; do not infer a second edit direction.
2. Start from the complete parent policy and implement the correction. Keep useful parent rules and preserve unrelated rules according to the supplied instruction and task constraints.
3. Trajectory examples are consumed upstream by the description agent and are intentionally absent here: implement the exact refined description.
4. Express the modification as concrete conditional policy rules. Return the complete resulting policy, not merely the changed rule or a checklist of required components.
5. The returned policy string is passed unchanged to the next-action model at every environment step. It is final natural-language policy text, not source code, a format template, or an object that another stage interpolates.
6. Across generated candidates, vary an actual conditional rule or policy mechanism consistent with the correction; paraphrases are not distinct artifacts.

FINAL CHECK
Confirm that the complete policy implements the supplied correction, preserves unaffected valid behavior, and follows S_C. Do not reproduce an excluded policy, introduce an unsupported edit direction, or invent measured performance.

Figure 9: Model Agent $\mathcal{G}_\phi$ prompt.

Single-objective Molecular Editing. SemanticRelaxation achieves the highest property-optimization accuracy in 9 of the 12 configurations in Tab. 6, improving average accuracy over AgentDrug by 4.2 and 2.9 percentage points under loose and strict criteria, respectively. The largest improvement occurs for +QED under loose criteria, where accuracy increases from 60.4% to 74.4%. Performance gains

Table 6: Single-objective Molecular editing results with Llama-3.1-8B. Validity and property-optimization accuracy are reported in percentage. l and s denote loose and strict settings, respectively.

Objective	d	Vanilla		REINVENT		ChatDrug		AgentDrug		SemanticRelaxation	
		Valid	Acc.	Valid	Acc.	Valid	Acc.	Valid	Acc.	Valid	Acc.
+LogP	l	76.3	40.1	73.3	32.9	76.0	48.3	**85.8**	69.1	70.8	**69.2**
	s	71.7	26.2	75.4	26.8	71.4	38.6	**81.6**	61.6	70.6	**68.4**
−LogP	l	82.0	25.8	**84.8**	33.5	75.8	36.4	83.0	56.8	80.2	**68.6**
	s	79.0	13.4	81.7	7.5	72.7	27.3	**87.0**	60.9	79.0	**68.0**
+TPSA	l	75.2	25.9	77.0	29.2	74.6	42.5	**85.8**	**70.4**	69.4	66.2
	s	74.9	24.0	74.7	25.6	74.6	44.0	**85.8**	**70.0**	68.8	65.6
−TPSA	l	78.4	16.5	79.2	17.2	74.9	41.6	**89.7**	75.8	82.4	**78.4**
	s	82.0	17.2	84.9	19.9	77.2	40.9	**86.6**	66.7	79.6	**72.4**
+QED	l	80.0	18.4	81.7	21.7	75.2	39.9	**85.1**	60.4	84.6	**74.4**
	s	80.6	2.8	80.0	4.7	74.6	17.9	**83.0**	45.6	71.6	**48.2**
−QED	l	77.5	31.8	76.1	29.3	73.0	39.4	**86.6**	68.8	75.4	**70.0**
	s	81.6	22.4	80.2	16.3	71.9	34.2	**85.5**	**66.2**	73.4	65.8
Average	l	78.2	26.4	78.7	27.0	74.9	41.3	**86.0**	66.9	77.1	**71.1**
	s	78.3	17.7	79.5	16.5	73.8	33.8	**84.9**	61.8	73.8	**64.7**

nevertheless vary across properties: AgentDrug remains stronger on $+\mathrm{TPSA}$ under both criteria and on $-\mathrm{QED}$ under strict criteria. Semantic refinement therefore benefits most evaluated objectives, while some property requirements remain challenging.

Multi-objective Molecular Editing. SemanticRelaxation achieves the highest accuracy across all 16 configurations in Tab. 7. Average accuracy reaches 60.3% and 50.9% under loose and strict criteria, exceeding AgentDrug by 29.0 and 29.4 percentage points, respectively. For $-\mathrm{LogP}-\mathrm{TPSA}$, accuracy improves from the strongest baseline's 9.7% to 64.6% under loose criteria and from 6.0% to 57.4% under strict criteria. Larger gains under combined objectives suggest that semantic refinement is particularly useful when multiple property requirements must be satisfied jointly. However, average molecular validity remains below AgentDrug in both single- and multi-objective settings, indicating that improved property optimization does not uniformly improve chemical validity.

Computational Efficiency. Tab. 8 compares the computational overhead of SemanticRelaxation and MaaS on MBPP and HumanEval. Training token consumption decreases by 14.85% and 23.82%, respectively, while inference token consumption decreases by 23.35% and 45.22%. Reductions occur in both prompt and completion tokens, showing savings in the context processed and the output generated. Total API costs decrease by 18.29% on MBPP and 31.31% on HumanEval, accompanied by wall-clock time reductions of 22.90% and 31.34%. Lower token consumption and shorter execution time across both benchmarks support the computational efficiency of SemanticRelaxation during training and inference.

Qualitative Results. We also presents qualitative results across benchmarks, as shown in Figs. 10 to 15. For AgentDrug, replacing a peripheral bromine atom with fluorine reduces LogP from 4.02 to 3.40 while preserving the molecular core. Simplifying the molecular framework in the second example decreases LogP from 3.18 to 0.01 and improves QED from 0.50 to 0.86. Fig. 12 illustrates a property trade-off: replacing an oxygen-containing group with nitrogen-containing functionality lowers LogP from 0.90 to -0.64, but also decreases QED from 0.88 to 0.70. These observations indicate that the effects of molecular edits depend on the property being evaluated. For NAS-Bench-201, refinement instructions specify changes to selected edge operations while preserving the remaining cell structure. The examples include introducing 3×3 convolutions and adjusting the placement of skip connections. The corresponding parent–child transitions increase accuracy from 89.83% to 91.02% on CIFAR-10, from 69.64% to 70.88% on CIFAR-100, and from 44.00% to 45.63% on ImageNet16-120, demonstrating improvements through localized architectural edits in these cases.

E LIMITATIONS

SemanticRelaxation requires prior training of the joint semantic representation, introducing additional computational overhead before artifact optimization. Future work could reduce this initial cost

Table 7: Multi-objective Molecular editing results with Llama-3.1-8B. Validity and property-optimization accuracy are reported in percentage. l and s denote loose and strict settings, respectively.

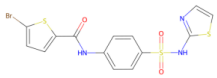
Objective	d	Vanilla		REINVENT		ChatDrug		AgentDrug		SemanticRelaxation	
		Valid	Acc.	Valid	Acc.	Valid	Acc.	Valid	Acc.	Valid	Acc.
+LogP + TPSA	ll	73.3	11.0	72.5	20.4	67.1	23.8	**73.8**	42.4	67.2	**59.0**
	ss	69.2	10.0	69.2	13.0	66.4	19.6	**71.7**	31.9	58.2	**48.8**
+LogP − TPSA	ll	76.3	11.8	74.7	14.1	71.2	19.6	**78.4**	27.1	71.4	**64.4**
	ss	74.6	6.0	74.9	6.7	70.4	8.8	**80.6**	16.1	71.4	**62.4**
−LogP + TPSA	ll	73.8	11.4	72.5	17.8	71.2	19.2	**78.4**	42.0	73.4	**58.6**
	ss	70.4	6.7	73.9	8.7	70.9	14.5	**75.5**	29.8	69.8	**56.2**
−LogP − TPSA	ll	77.5	3.1	**81.6**	3.0	71.9	9.7	79.7	9.6	80.0	**64.6**
	ss	76.3	0.8	79.3	0.0	73.0	5.1	**80.6**	6.0	76.4	**57.4**
+LogP + QED	ll	81.0	13.4	**84.9**	15.9	73.0	20.4	83.0	46.0	70.6	**59.4**
	ss	80.0	3.2	**81.3**	3.3	71.2	7.5	74.6	23.9	69.0	**42.6**
+LogP − QED	ll	76.0	25.9	72.6	20.1	74.9	28.5	**77.2**	35.5	65.4	**59.2**
	ss	86.2	15.5	**88.5**	17.1	81.0	22.7	75.8	16.3	58.2	**53.6**
−LogP + QED	ll	78.4	8.2	**81.7**	11.9	77.2	21.6	77.5	32.9	74.6	**58.4**
	ss	77.8	1.2	**80.3**	0.9	76.0	7.2	78.1	21.5	74.4	**39.2**
−LogP − QED	ll	77.8	13.2	77.3	14.3	75.5	14.3	**81.3**	14.6	76.0	**58.6**
	ss	83.0	24.5	82.7	7.3	78.1	22.6	**86.2**	26.3	69.0	**46.6**
Average	ll	76.8	12.3	77.1	13.8	72.7	19.6	**78.7**	31.3	72.3	**60.3**
	ss	77.2	8.5	**78.8**	8.3	73.4	13.5	77.9	21.5	68.3	**50.9**

Table 8: **Efficiency comparison on MBPP and HumanEval.** Tokens and time are reported in millions (M) and hours (h).

Category	Component	MBPP		HumanEval	
		MaAS	SemanticRelaxation	MaAS	SemanticRelaxation
Training	Prompt	38.30M	**33.06M** (-13.68%)	7.83M	**6.10M** (-22.07%)
	Completion	7.69M	**6.10M** (-20.65%)	2.10M	**1.46M** (-30.32%)
	Total	45.99M	**39.16M** (-14.85%)	9.92M	**7.56M** (-23.82%)
Inference	Prompt	11.80M	**9.03M** (-23.49%)	2.57M	**1.41M** (-44.86%)
	Completion	2.43M	**1.88M** (-22.67%)	0.76M	**0.41M** (-46.41%)
	Total	14.23M	**10.91M** (-23.35%)	3.32M	**1.82M** (-45.22%)
Cost (USD)	Training	$10.36	**$8.62** (-16.79%)	$2.43	**$1.79** (-26.34%)
	Inference	$3.23	**$2.48** (-23.12%)	$0.84	**$0.46** (-45.70%)
	Total	$13.58	**$11.10** (-18.29%)	$3.27	**$2.25** (-31.31%)
Wall-clock time	Training	37.69h	**29.39h** (-22.03%)	6.67h	**4.65h** (-30.18%)
	Inference	12.59h	**9.38h** (-25.49%)	2.20h	**1.43h** (-34.86%)
	Total	50.28h	**38.76h** (-22.90%)	8.87h	**6.09h** (-31.34%)

through more efficient representation learning and investigate the reuse of learned representations across tasks.

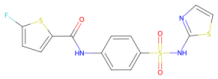
Parent Artifact (LogP: 4.02)



Parent Description

O=C(Nc1ccc(S(=O)(=O)Nc2nccs2)cc1)c1ccc(Br)s1

Child Artifact (LogP: 3.40)



Child Description

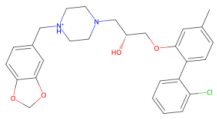
O=C(Nc1ccc(S(=O)(=O)Nc2nccs2)cc1)c1ccc(F)s1

Edit Motif

Replace one peripheral Br-containing site with F-containing functional character while retaining the parent core.

Figure 10: Example 1 on AgentDrug.

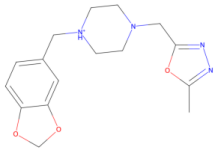
Parent Artifact (LogP: 3.18 · QED: 0.50)



Parent Description

Cc1ccc(-c2ccccc2Cl)c(OC[C@H](O)CN2CC[NH+](Cc3ccc4c(c3)OCO4)CC2)c1

Child Artifact (LogP: 0.01 · QED: 0.86)



Child Description

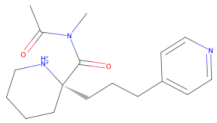
Cc1nnc(CN2CC[NH+](Cc3ccc4c(c3)OCO4)CC2)o1

Edit Motif

Remove one ring connection and simplify the molecular framework while retaining unaffected parent character.

Figure 11: Example 2 on AgentDrug.

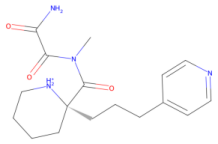
Parent Artifact (LogP: 0.90 · QED: 0.88)



Parent Description

CC(=O)N(C)C(=O)[C@@]1(CCCc2ccncc2)CCCC[NH2+]1

Child Artifact (LogP: -0.64 · QED: 0.70)



Child Description

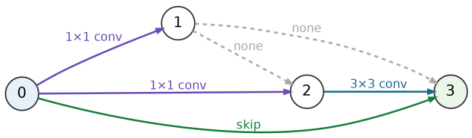
CN(C(=O)C(N)=O)C(=O)[C@@]1(CCCc2ccncc2)CCCC[NH2+]1

Edit Motif

Replace one peripheral C-containing site with N-containing functional character while retaining the parent core.

Figure 12: Example 3 on AgentDrug.

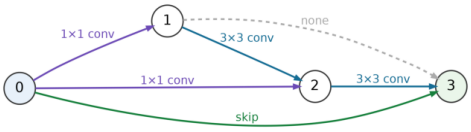
Parent Artifact (89.83%)



Parent Description

Start from this metric-blind CIFAR-10 NAS-Bench-201 cell: edge 0->1 uses nor_conv_1x1 to mix channels with low spatial cost; edge 0->2 uses nor_conv_1x1 to mix channels with low spatial cost; edge 1->2 uses none to remove a redundant computation path; edge 0->3 uses skip_connect to preserve information through a direct residual path; edge 1->3 uses none to remove a redundant computation path; edge 2->3 uses nor_conv…

Child Artifact (91.02%)



Child Description

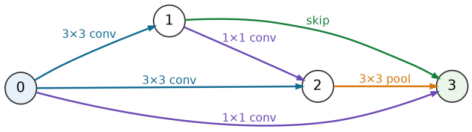
[ARCHITECTURE EDIT] edge 1->2: none -> nor_conv_3x3; preserve every other edge.

Edit Motif

edge 1->2: none -> nor_conv_3x3

Figure 13: CIRAR10 example on NAS-Bench-201

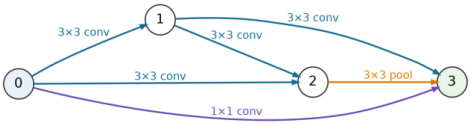
Parent Artifact (69.64%)



Parent Description

Start from this metric-blind CIFAR-100 NAS-Bench-201 cell: edge 0->1 uses nor_conv_3x3 to capture local spatial structure with a learned operator; edge 0->2 uses nor_conv_3x3 to capture local spatial structure with a learned operator; edge 1->2 uses nor_conv_1x1 to mix channels with low spatial cost; edge 0->3 uses nor_conv_1x1 to mix channels with low spatial cost; edge 1->3 uses skip_connect to preserve informatio…

Child Artifact (70.88%)



Child Description

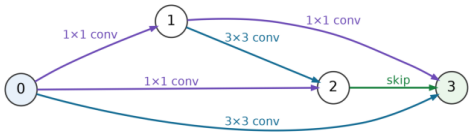
[ARCHITECTURE EDIT] edge 1->2: nor_conv_1x1 -> nor_conv_3x3; edge 1->3: skip_connect -> nor_conv_3x3; preserve every other edge.

Edit Motif

edge 1->2: nor_conv_1x1 -> nor_conv_3x3; edge 1->3: skip_connect -> nor_conv_3x3

Figure 14: CIRAR100 example on NAS-Bench-201

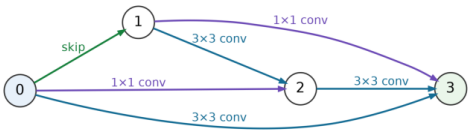
Parent Artifact (44.00%)



Parent Description

Start from this metric-blind ImageNet16-120 NAS-Bench-201 cell: edge 0->1 uses nor_conv_1x1 to mix channels with low spatial cost; edge 0->2 uses nor_conv_1x1 to mix channels with low spatial cost; edge 1->2 uses nor_conv_3x3 to capture local spatial structure with a learned operator; edge 0->3 uses nor_conv_3x3 to capture local spatial structure with a learned operator; edge 1->3 uses nor_conv_1x1 to mix channels w…

Child Artifact (45.63%)



Child Description

[ARCHITECTURE EDIT] edge 0->1: nor_conv_1x1 -> skip_connect; edge 2->3: skip_connect -> nor_conv_3x3; preserve every other edge.

Edit Motif

edge 0->1: nor_conv_1x1 -> skip_connect; edge 2->3: skip_connect -> nor_conv_3x3

Figure 15: ImageNet16-120 example on NAS-Bench-201