SGEvolve: Semantic Gradient Guidance for LLM-Driven Evolutionary Search

Kewei Feng[1,3] **Jinbiao Nie**[2,3] **Xiaoyuan Zhang**[3*] **Quanhua Liu**[4]
[1]School of Information and Electronics, Beijing Institute of Technology
[2]School of Electronic Engineering, Beijing University of Posts and Telecommunications
[3]Zhongguancun Academy
[4]Chongqing Innovation Center, Beijing Institute of Technology

Abstract

Large language model (LLM)-driven evolutionary frameworks have emerged as a promising paradigm for iterative search in complex optimization and structured generation tasks. However, existing methods typically rely on heuristic variation operators, leaving directional information in observed performance feedback underexploited and limiting both efficiency and reliability. To address this, we propose SGEvolve, a semantic gradient-guided evolutionary framework for LLM-driven optimization. The key idea is to treat the performance differences between parent and offspring as zeroth-order evidence of an underlying ascent direction. Based on this view, we formulate semantic gradient extraction as a Maximum A Posteriori (MAP) estimation problem, where the LLM infers the most probable direction of improvement conditioned on observed evolutionary steps. The inferred gradients are then used to guide mutation and crossover, biasing generation toward more promising regions of the search space. We evaluate SGEvolve on a diverse set of tasks, including mathematical optimization, algorithm design, and real-world industrial problems. Experimental results demonstrate that SGEvolve consistently outperforms existing open-source baselines, achieving superior performance in both solution quality and search efficiency. Our code is available at **SGEvolve**.

1 Introduction

Recent advancements in Large Language Models (LLMs) are increasingly shifting the focus of scaling from training-time compute to inference-time computation [1–3], where performance gains are achieved through more deliberate and structured reasoning processes [4–6]. This transition has catalyzed a move from static content generation to dynamic, iterative problem-solving in complex domains such as mathematical discovery and program generation [7–9]. Within this context, LLMs are no longer merely producing outputs; they are functioning as intelligent search agents capable of exploring vast solution spaces through continuous refinement. This trend is most evident in the rise of LLM-driven iterative frameworks, which treat the model as a heuristic operator within an evolutionary-style search process [10]. By iteratively evaluating, selecting, and modifying candidates, these frameworks attempt to traverse the optimization landscape with greater precision than traditional stochastic methods. This shift positions LLMs as the foundational infrastructure for iterative search, setting the stage for more sophisticated mechanisms that can navigate these spaces with explicit directional guidance and improved efficiency.

Current LLM-driven evolutionary frameworks typically establish an iterative closed-loop: a population of candidates is maintained, refined via LLM-facilitated operators, and filtered through external

*Corresponding author. E-mail: zhangxiaoyuan@bza.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

fitness feedback. Within this paradigm, existing research can be broadly divided into two categories. One approach utilizes LLMs as static heuristic optimizers, relying on meticulously designed prompts to harness the model's zero-shot or few-shot synthesis capabilities [11–13]. Alternatively, the other approach explores model-in-the-loop evolution, where parameters are updated via Reinforcement Learning (RL) to internalize the optimization logic [14–16]. The prominence of these evolutionary frameworks stems from their unique capacity for black-box optimization; by maintaining a diverse population and employing stochastic variation, they circumvent the limitations of traditional methods in discrete or poorly-defined landscapes, achieving significant breakthroughs across a wide array of tasks.

Despite their promising potential, existing LLM-driven evolutionary frameworks often encounter bottlenecks in sampling efficiency and stochastic stability. While these frameworks excel at broad exploration and often incorporate empirical feedback to guide mutations or crossovers [17–19], such mechanisms predominantly rely on heuristic-driven prompting. Without an explicit mechanism to systematically map fitness variations to specific improvement trajectories, the LLM struggles to understand how and why certain modifications lead to better fitness. Consequently, searching through vast semantic spaces remains heavily reliant on stochastic trial and error, requiring extensive evaluations to reach convergence and often stagnating in local optima. This motivates a *structural anchor* for semantic search: a directional signal grounded in evaluated modifications. Just as gradients serve as a compass in mathematical optimization, such a signal can turn observed fitness changes into actionable guidance for navigating complex semantic landscapes.

We hypothesize, however, that the essence of a gradient—the signal of steepest ascent—is not entirely lost in non-differentiable spaces, but rather remains latent, a notion aligned with the finding that Transformers can implicitly perform gradient descent as meta-optimizers [20, 21]. We observe that the performance differences between parent and offspring solutions implicitly encode vital directional information regarding the optimization landscape. This leads to a pivotal realization: while explicit mathematical derivatives are inaccessible, these performance differences can be decoded as "semantic gradients" by the LLM itself.

Based on this idea, we propose SGEvolve, a framework that integrates semantic gradient extraction with evolutionary operators. Specifically, SGEvolve comprises three core components: semantic gradient extraction, crossover, and mutation. The semantic gradient extraction operator interprets the observed performance differences between parent and offspring as zeroth-order evidence of an underlying ascent direction in a latent continuous space conceptually induced by the LLM. Concretely, it formulates the recovery of this direction as a Maximum A Posteriori estimation problem, where the LLM infers a textual rule whose semantic embedding best aligns with the observed performance variation, effectively approximating the latent gradient direction. Conditioned on the inferred semantic gradient, the crossover and mutation operators move beyond purely stochastic variation. Instead, they bias the generation process toward candidates that follow the inferred direction, resulting in more targeted exploration of the search space. Under smoothness and suitably bounded sampling noise, our analysis shows that such directional bias yields positive expected improvement when the inferred semantic gradients satisfy a positive-alignment condition. We summarize our main contributions as follows.

- **Formulation of Semantic Gradients.** We formulate semantic gradients from evaluated parent–offspring transitions, bridging a key gap between classical gradient-based optimization and LLM-driven evolutionary search. By decoding the performance differences between parent and offspring solutions into directional signals, we demonstrate how LLMs can approximate gradient-like guidance to navigate discrete, non-differentiable semantic spaces.
- **The SGEvolve Framework.** We propose SGEvolve, a novel LLM-driven evolutionary framework that operationalizes semantic gradients for directed search. The framework comprises three core operators—semantic gradient extraction, crossover, and mutation—where the inferred semantic gradients explicitly guide the generation of new candidates. This design transforms conventional heuristic evolution into a more directed and efficient search process.
- **Empirical Effectiveness Across Diverse Domains.** We conduct extensive experiments on a diverse set of challenging tasks, including mathematical optimization, algorithm design, and real-world industrial problems. The results show that SGEvolve consistently outperforms

strong open-source baselines, achieving improvements in both solution quality and search efficiency.

2 Related Works

LLM-Driven Evolutionary Search. Recent work has increasingly explored large language models (LLMs) as engines for population-based search, giving rise to a new class of LLM-driven evolutionary frameworks. In these approaches, candidate solutions are iteratively refined through LLM-driven variation operators, such as mutation and crossover, while external feedback signals are used to evaluate and select promising candidates.

Early studies, including FunSearch [7] and EoH [11], pioneered the paradigm of treating LLMs as semantic operators for evolutionary search. Building on this foundation, AlphaEvolve [9] introduced more structured evolutionary components, such as population management, crossover, and mutation, extending the applicability of LLM-driven evolution to scientific discovery and engineering tasks. Subsequent works have focused on improving the efficiency and stability of the evolutionary process. For instance, ReEvo [22] and HiFoPrompt [18] leverage historical trajectories to extract useful experience for guiding future evolution. ShinkaEvolve [12] improves sample efficiency through enhanced parent selection, rejection-sampled code rewriting, and adaptive LLM ensembles. PACEvolve [23] addresses challenges such as context contamination, mode collapse, and inefficient collaboration, aiming to enable stable long-horizon evolution. Complementary efforts, such as DeltaEvolve [24], explore more efficient utilization of contextual information within the evolutionary loop. Beyond context management, other works investigate algorithmic control over the evolutionary process. AdaEvolve [13] introduces adaptive control strategies to optimize LLM-driven program synthesis.

In parallel, several approaches incorporate reinforcement learning by treating evaluation signals as rewards, thereby internalizing task-specific optimization strategies into model parameters [14, 15]. While effective, such methods typically incur substantially higher computational costs due to parameter updates.

Gradient-like Guidance in LLM-Based Optimization. A growing body of work investigates whether LLMs can emulate gradient-based optimization in non-differentiable semantic spaces. ProTeGi [25] translates the idea of numerical gradient descent into discrete text optimization by using natural language feedback generated by LLMs as "textual gradients," enabling iterative prompt refinement through semantic edits. Building heavily upon this direction, TextGrad [26] introduces a powerful framework that backpropagates textual feedback provided by LLMs to optimize individual components within compound AI systems, treating text suggestions as gradients in a computation graph. Similarly, GPO [27] draws an analogy to gradient descent by retrieving historical trajectories to infer optimization directions, combined with controlled generative refinement for precise prompt updates. Other approaches adopt more implicit forms of gradient-like guidance. For example, Reflexion [28] enables models to generate verbal feedback that constrains future actions. While these methods demonstrate the potential of extracting gradient-like information from language models, they typically focus on optimizing single trajectories or computation graphs. Distinct from TextGrad and similar methods, our approach marries this gradient-like extraction with population-based evolutionary search.

3 Method

To address the absence of explicit directional signals in discrete semantic spaces, we introduce the concept of "semantic gradients". In this section, we construct a conceptual mathematical framework that maps discrete text into a continuous latent space, bridging heuristic evolutionary search with a zeroth-order optimization analogy. An overview of the proposed framework is illustrated in Figure 1.

3.1 Semantic Gradient Formulation

Let the discrete search space be $\mathcal{S}$ and the objective function be $\mathcal{F} : \mathcal{S} \to \mathbb{R}$. We assume the existence of a continuous latent representation space $\mathcal{Z} \subset \mathbb{R}^d$ induced by the LLM, alongside a mapping $\phi : \mathcal{S} \to \mathcal{Z}$. Accordingly, we define a continuous surrogate fitness function $f : \mathcal{Z} \to \mathbb{R}$ such that:

$$\mathcal{F}(s) = f(\phi(s)) \tag{1}$$

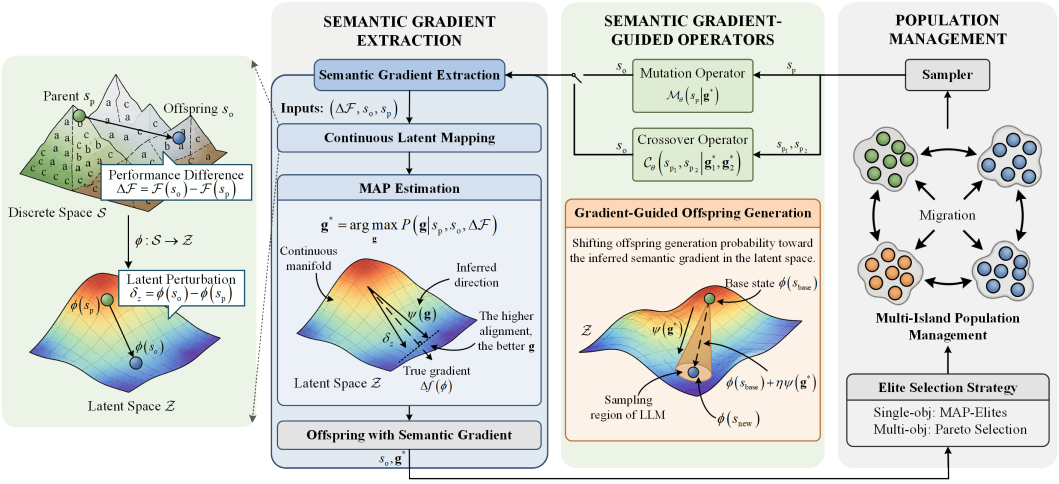


Figure 1: Overview of the SGEvolve framework. By mapping the discrete semantic space to a continuous latent space (Left), SGEvolve translates parent-offspring performance differences into directional signals. The framework extracts a semantic gradient $\mathbf{g}^*$ from evaluated transitions, with MAP estimation characterizing the inference of an improvement direction (Center-Left). This inferred gradient is then utilized by the mutation and crossover operators to bias the offspring sampling distribution toward promising regions (Center-Right), operating within a robust multi-island population management system (Right).

During evolution, consider a parent s_p and its generated offspring s_o. Assuming f is locally differentiable and the evolutionary mutation step is sufficiently small, the observed fitness difference $\Delta\mathcal{F} = \mathcal{F}(s_\mathrm{o}) - \mathcal{F}(s_\mathrm{p})$ can be approximated via a first-order Taylor expansion:

$$\Delta\mathcal{F} \approx \langle \nabla f(\phi(s_\mathrm{p})), \delta_z \rangle = \nabla f(\phi(s_\mathrm{p}))^T (\phi(s_\mathrm{o}) - \phi(s_\mathrm{p})) \tag{2}$$

where $\delta_z = \phi(s_\mathrm{o}) - \phi(s_\mathrm{p})$ denotes the latent perturbation induced by the evolutionary step. This relation reveals that the observable fitness change $\Delta\mathcal{F}$ is directly governed by the inner product of the latent perturbation δ_z and the unobserved steepest-ascent direction ∇f.

Since ∇f is not directly accessible in the discrete semantic domain, our goal is to recover a textual descriptor $\mathbf{g} \in \mathcal{V}$, where $\mathcal{V}$ denotes the space of valid natural language instructions, such that the semantic embedding $\psi(\mathbf{g})$ approximates the true ascent direction. We refer to $\mathbf{g}$ as the *semantic gradient*.

3.2 Semantic Gradient Extraction and Gradient-Guided Operators

We conceptually formulate the LLM's Semantic Gradient Extraction operator, $\mathcal{E}_\theta$, through the lens of a MAP estimation problem. It is crucial to note that the continuous latent space $\mathcal{Z}$ and the mapping functions ϕ, ψ serve as a conceptual mathematical framework to interpret the LLM's internal semantic representations, rather than explicit numerical vectors computed during the search. Given the observed evolutionary step $(s_\mathrm{p}, s_\mathrm{o})$ and the corresponding fitness change $\Delta\mathcal{F}$, we seek the most probable directional textual rule $\mathbf{g}^*$:

$$\mathbf{g}^* = \arg\max_{\mathbf{g}} P(\mathbf{g} \mid s_\mathrm{p}, s_\mathrm{o}, \Delta\mathcal{F}) \tag{3}$$

Using Bayes' theorem, the posterior can be decomposed into a likelihood term and a prior term:

$$P(\mathbf{g} \mid s_\mathrm{p}, s_\mathrm{o}, \Delta\mathcal{F}) \propto P(\Delta\mathcal{F} \mid \mathbf{g}, s_\mathrm{p}, s_\mathrm{o}) \cdot P(\mathbf{g} \mid s_\mathrm{p}, s_\mathrm{o}) \tag{4}$$

The likelihood term models how well the semantic direction $\mathbf{g}$ accounts for the observed fitness variation. Within this conceptual model, we characterize the likelihood by how closely the observed $\Delta\mathcal{F}$ matches the expected variation in the latent space. Specifically, we formulate this likelihood as a Gaussian distribution, which peaks when $\Delta\mathcal{F}$ perfectly aligns with the inner product between the inferred direction $\psi(\mathbf{g})$ and the latent step δ_z:

$$P(\Delta\mathcal{F} \mid \mathbf{g}, s_\mathrm{p}, s_\mathrm{o}) \propto \exp\left(-\frac{1}{2\sigma^2}\Big(\Delta\mathcal{F} - \langle \psi(\mathbf{g}), \delta_z \rangle\Big)^2\right) \tag{5}$$

Intuitively, if the projection of the perturbation δ_z onto the inferred direction $\psi(\mathbf{g})$ correlates well with $\Delta\mathcal{F}$, the likelihood becomes larger. The prior $P(\mathbf{g} \mid s_{\mathrm{p}}, s_{\mathrm{o}})$ is provided by the LLM's pre-trained world knowledge, ensuring that $\mathbf{g}$ is a logically sound and semantically coherent strategy.

By prompting the LLM with the inputs $(s_{\mathrm{p}}, s_{\mathrm{o}}, \Delta\mathcal{F})$, the extraction operator $\mathcal{E}_\theta$ simulates this estimation and outputs a textual rule $\mathbf{g}^*$ (e.g., *"Start the optimization with slightly relaxed non-overlap constraints."*). In this way, the LLM serves as an engine that transforms observed performance differences into an explicit semantic direction for search.

With the necessary semantic gradients derived, we further construct semantic gradient-guided mutation and crossover operators. For mutation, the new candidate is generated as

$$s_{\text{new}} = \mathcal{M}_\theta(s_{\mathrm{p}} \mid \mathbf{g}^*).$$

For crossover, the new candidate is generated from two parents as

$$s_{\text{new}} = \mathcal{C}_\theta(s_{\mathrm{p}_1}, s_{\mathrm{p}_2} \mid \mathbf{g}_1^*, \mathbf{g}_2^*).$$

We view both processes as sampling from a distribution concentrated around a semantic target translated along the inferred gradient direction. Let ϕ_{base} denote the base latent state, where

$$\phi_{\text{base}} = \phi(s_{\mathrm{p}})$$

for mutation, and

$$\phi_{\text{base}} = \beta\phi(s_{\mathrm{p}_1}) + (1-\beta)\phi(s_{\mathrm{p}_2}), \quad \beta \in (0,1),$$

where $\beta \in (0,1)$ is an interpolation weight, for crossover. The probability of generating s_{new} is then written as

$$P(s_{\text{new}} \mid \phi_{\text{base}}, \mathbf{g}^*) \propto \exp\Big(-\lambda\|\phi(s_{\text{new}}) - (\phi_{\text{base}} + \eta\psi(\mathbf{g}^*))\|^2\Big), \tag{6}$$

where $\eta > 0$ denotes the semantic step size controlled adaptively by the LLM, and λ controls the concentration of the generation distribution.

Motivation for Expected Improvement. The above formulation establishes a connection between zeroth-order optimization and evolutionary search. Under a positive-alignment assumption, semantic-gradient guidance introduces a favorable first-order drift in the surrogate fitness. The following proposition formalizes when this drift outweighs the effects of curvature and sampling noise, providing a conditional sufficient-condition result for expected improvement.

Proposition 1 (Expected Fitness Improvement under Positive Alignment)**.** *Assume the continuous surrogate function f is L-smooth with $L > 0$. At a fixed base state ϕ_{base} with a nonzero gradient, suppose the inferred semantic direction satisfies*

$$\mathbb{E}\left[\langle \nabla f(\phi_{base}), \psi(\mathbf{g}^*)\rangle\right] \geq \mu\|\nabla f(\phi_{base})\|^2, \qquad \mu > 0. \tag{7}$$

Under the zero-mean noise assumptions specified in Appendix A, the guided step yields positive expected surrogate fitness improvement for a step size and noise bound satisfying Equation 12.

The detailed proof and further theoretical discussion are provided in Appendix A, and the complete prompt templates and an example of semantic gradient extraction are provided in Appendix C. Complementing this conditional analysis, Section 4.5 examines the consistency between extracted directions and subsequent fitness changes during actual search.

3.3 Population Management and Sampling Strategy

To support stable and scalable evolution, SGEvolve maintains a population database that stores generated candidates throughout the search process. The framework adopts a multi-island evolution strategy, where multiple sub-populations evolve independently for a fixed number of iterations before exchanging elite individuals periodically [9, 29]. This design helps preserve diversity while reducing the risk of premature convergence.

SGEvolve supports both single-objective and multi-objective optimization. For single-objective problems, we use the MAP-Elites algorithm to select the most promising candidates [30]. For multi-objective problems, we adopt Pareto selection to retain a set of non-dominated elites. In both cases, the elite set serves as the anchor for organizing the search state and guiding information exchange across islands.

Algorithm 1 SGEvolve: Semantic Gradient-Guided Evolution

Require: Task specification $\mathcal{T}$, objective function $\mathcal{F}$, LLM-based operators $(\mathcal{E}_\theta, \mathcal{M}_\theta, \mathcal{C}_\theta)$, iteration budget B, population hyperparameters $\mathcal{H}$
Ensure: The best-found candidate s^* (single-objective) or Pareto front $\mathcal{P}^*$ (multi-objective)

Generate initial candidates from task specification $\mathcal{T}$ and population hyperparameters $\mathcal{H}$
Evaluate all initial candidates with $\mathcal{F}$ and assign heuristic initial semantic gradients; populate database $\mathcal{D}$ across islands
while iteration budget B is not exhausted **do**
    Sample operator $op \in \{\text{Mutation}, \text{Crossover}\}$ based on fixed ratio
    if op is Mutation **then**
        Sample parent s_{p} and its associated semantic gradient $\mathbf{g}_p^*$ from $\mathcal{D}$
        Generate offspring $s_{\text{o}} \leftarrow \mathcal{M}_\theta(s_{\text{p}} \mid \mathbf{g}_p^*)$ ▷ Gradient-guided mutation
        Set base reference $s_{\text{base}} \leftarrow s_{\text{p}}$
    else
        Sample parents $(s_{\text{p}_1}, s_{\text{p}_2})$ and their gradients from the elite set $\mathcal{D}_{\text{elite}}$
        Generate offspring $s_{\text{o}} \leftarrow \mathcal{C}_\theta(s_{\text{p}_1}, s_{\text{p}_2} \mid \mathbf{g}_{p_1}^*, \mathbf{g}_{p_2}^*)$ ▷ Gradient-guided recombination
        Set base reference s_{base} by aggregating s_{p_1} and s_{p_2}
    end if
    Evaluate fitness $\mathcal{F}(s_{\text{o}})$
    Compute performance differences $\Delta\mathcal{F} = \mathcal{F}(s_{\text{o}}) - \mathcal{F}(s_{\text{base}})$
    Extract new semantic gradient: $\mathbf{g}_o^* \leftarrow \mathcal{E}_\theta(s_{\text{base}}, s_{\text{o}}, \Delta\mathcal{F})$ ▷ MAP Estimation via LLM
    Insert tuple $(s_{\text{o}}, \mathcal{F}(s_{\text{o}}), \mathbf{g}_o^*)$ into database $\mathcal{D}$
    Update the elite set $\mathcal{D}_{\text{elite}}$ via MAP-Elites selection or Pareto sorting
    if island exchange condition is triggered **then**
        Migrate elite individuals across sub-populations to preserve diversity
    end if
end while
return Best solution(s) from the elite set $\mathcal{D}_{\text{elite}}$

During evolution, SGEvolve samples mutation or crossover with a fixed ratio. Since the two operators play different roles, we design distinct sampling scopes for them. For mutation, the sampling range covers the entire database, allowing the operator to explore a broader region of the search space and introduce diversity. For crossover, the sampling range is restricted to elite candidates. This is because crossover combines two parent solutions, and its effectiveness depends on the compatibility of their latent representations. If the two parents are too far apart, the interpolated latent state may become unstable and less informative. By sampling crossover parents from the elite set, SGEvolve improves the stability of recombination while better exploiting high-quality candidates and accelerating convergence.

Algorithm 1 summarizes the overall procedure of SGEvolve.

4 Experiments

We evaluate SGEvolve on 115 diverse problems across three domains: mathematical optimization (5 tasks, Section 4.2), algorithm design (10 tasks, Section 4.3), and real-world industrial optimization (100 tasks, Section 4.4). Finally, Section 4.5 examines the contributions of individual operators, the benefit of directional guidance beyond reflection, and its consistency with subsequent fitness changes.

4.1 Experimental Setup

Main experiments are conducted based on the Gemini-3-Flash backbone [31]. For each problem, all methods share the same evaluator to ensure consistent fitness assessment.

Baselines and Computational Budget Fairness. We compare SGEvolve against open-source state-of-the-art frameworks discussed in Section 2: ReEvo, HiFoPrompt, AdaEvolve, and ShinkaEvolve. Since baselines adopt varying sampling and evolutionary strategies, directly comparing iterations or population sizes is inequitable. To ensure fair comparison, we standardize the computational budget

Table 1: Mathematical optimization results (mean ± std and best objective values) on five benchmarks. **Bold** denotes the best result among all methods. In cases where both the mean and best objective values are identical, the method with the lower standard deviation is regarded as better.

	Circle packing ↑		7D sphere packing ↑		Heilbronn (triangle) ↑		Moving sofa ↑		Max to min ratios ↓	
Method	Mean ± std	Best	Mean ± std	Best	Mean ± std	Best	Mean ± std	Best	Mean ± std	Best
Known best	–	2.636	–	–	–	0.0365	–	2.220	–	3.590
ReEvo	2.541 ± .000	2.541	7.707 ± .072	7.768	0.0340 ± .0023	0.0354	2.216 ± .006	2.220	3.590 ± .000	3.590
HiFoPrompt	2.634 ± .002	2.636	7.730 ± .081	7.793	0.0328 ± .0016	0.0345	1.920 ± .219	2.173	3.593 ± .006	3.590
AdaEvolve	2.635 ± .001	2.636	7.714 ± .126	7.823	0.0365 ± .0000	0.0365	2.219 ± .002	2.220	3.590 ± .000	3.590
ShinkaEvolve	2.630 ± .004	2.634	7.713 ± .057	7.777	0.0340 ± .0022	0.0365	2.204 ± .019	2.216	3.596 ± .008	3.592
SGEvolve	**2.636** ± .000	**2.636**	**7.837** ± .020	**7.865**	**0.0365 ± .0000**	**0.0365**	**2.220** ± .001	**2.221**	**3.590** ± .000	**3.590**

Table 2: Performance comparison on ALE-Bench LITE.

Task	`ahc008`	`ahc011`	`ahc015`	`ahc016`	`ahc024`	`ahc025`	`ahc026`	`ahc027`	`ahc039`	`ahc046`	**Mean score**
Initial score	1173	1237	1454	1445	1324	1228	2584	1223	2054	1872	1559.4
ReEvo	1173	1582	1518	1445	1324	1288	2637	1223	2054	1872	1611.6
HiFoPrompt	1173	1367	1454	1445	1324	1288	2584	1223	2054	1872	1578.4
AdaEvolve	1190	1460	1483	1623	1324	1262	2637	1223	2113	**2409**	1672.4
ShinkaEvolve	1208	**1670**	1454	1445	1324	**1432**	2725	1303	2054	2323	1693.8
SGEvolve	**1213**	1604	**2179**	**1644**	**1967**	1277	**2891**	**1833**	**2244**	2135	**1898.7**

by enforcing a strict cost limit. Budget configurations and SGEvolve hyperparameters are detailed in Appendix B.1.

4.2 Mathematical Optimization

Table 1 presents five representative mathematical optimization problems, spanning continuous optimization, combinatorial geometry, and optimization with structural constraints [32, 33]. These tasks feature diverse search space topologies and constraint complexities, offering a comprehensive benchmark for evaluating generalization across varying landscapes. Averaged over three independent runs per problem, SGEvolve matches or outperforms existing baselines in mean objective value across all five tasks. These results demonstrate the effectiveness of semantic-gradient guidance for optimization without explicit mathematical derivatives.

This consistent improvement is underpinned by explicit mechanisms depending on the problem structure. In circle packing and 7D sphere packing, complex constraints and dense local optima often stall traditional perturbation-based evolutionary methods. In contrast, by extracting latent improvement directions from performance differences via semantic gradients, SGEvolve navigates the continuous space along a highly directed trajectory, significantly accelerating convergence.

Point configuration problems (e.g., Heilbronn triangle and max-to-min ratios) are highly sensitive to global topological structures, where moving a single point often triggers cascading constraint violations, rendering local fine-tuning ineffective. SGEvolve excels in these scenarios by leveraging the LLM's prior knowledge during semantic gradient extraction. Instead of treating local coordinates in isolation, SGEvolve captures global structural signals across individuals to infer adjustments for the joint distribution of the point set. This enables the search process to bypass degenerate configurations and steadily evolve toward superior topological structures.

Finally, in the moving sofa problem, the optimization must maximize area (a continuous objective) while navigating a right-angled corridor (an implicit structural constraint). While traditional methods frequently generate invalid samples in such heavily constrained spaces, SGEvolve achieves higher numerical area estimates under the shared evaluator, with the reported values subject to approximation errors in the construction and evaluation of the sofa. Semantic-gradient guidance helps incorporate geometric constraints into candidate generation.

4.3 ALE-Bench

While Section 4.2 validates the capability of SGEvolve to perform optimization from scratch, we now focus on refining suboptimal solutions within discrete and fragile algorithmic spaces using

Table 3: Performance comparison on IndustryOR benchmark in terms of weighted score and accuracy.

Method	Score	Accuracy
ReEvo	149	0.78
HiFoPrompt	142	0.75
AdaEvolve	186	0.95
ShinkaEvolve	180	0.94
SGEvolve	**187**	**0.95**

Table 4: Ablation results on circle packing and Heilbronn (triangle) problems. We report the mean, standard deviation and best of 3 runs

	Circle packing		Heilbronn (triangle)	
Ablation setting	Mean ± std	Best	Mean ± std	Best
SGEvolve	**2.635981** ± .000003	**2.635983**	**0.036530** ± .000000	**0.036530**
w/o Semantic Gradient Extraction	2.630432 ± .001040	2.631090	0.034095 ± .000728	0.034906
w/o Crossover	2.634425 ± .001228	2.635328	0.035187 ± .001092	0.036267
w/o Mutation	2.633035 ± .002486	2.635881	0.035330 ± .001465	0.036530

ALE-Bench LITE [34]. This benchmark comprises 10 challenging heuristic programming tasks from AtCoder, designed to assess LLMs in complex algorithm design. We initialize the population with the best solution generated by ALE-Agent (built upon Gemini-3-Flash) and evolve all methods from this common starting point. Detailed results are reported in Table 2.

SGEvolve achieves the highest score on 7 out of 10 tasks and surpasses the strongest baseline by over 200 points in average score across all tasks, demonstrating a substantial advantage in evolutionary optimization. Notably, except for the *ahc011* task, several baselines return final solutions identical to their initialization; on *ahc024*, all baselines fail to improve fitness entirely. This phenomenon of *evolutionary stagnation* highlights the limitations of unguided or weakly guided evolutionary frameworks in complex algorithmic spaces.

In contrast, SGEvolve effectively breaks this stagnation, achieving clear improvements across all tasks. We attribute this success to the guidance provided by the semantic gradient extraction operator. Instead of relying on blind modifications of parameters or code structures, SGEvolve analyzes structural and fitness differences between parent and offspring solutions, implicitly performing posterior inference to uncover the underlying evolutionary direction. The resulting semantic gradient serves as a logically grounded directional signal, guiding the mutation and crossover operators toward more promising regions of the search space. The improvements across all ten tasks support the effectiveness of this directional guidance in advancing the population beyond its initial solutions.

4.4 Real-World Industrial Optimization

Finally, we evaluate SGEvolve on IndustryOR [35], a benchmark comprising 100 real-world industrial optimization problems. To account for varying complexities, tasks are categorized by difficulty—Easy (40), Medium (40), and Hard (20)—awarding +1, +2, and +4 points respectively for correct solutions. Table 3 summarizes the aggregate scores and overall correctness rates.

SGEvolve achieves the highest overall score while tying for first place in the unweighted correctness rate. This specific distribution of results indicates that SGEvolve is particularly effective at resolving the Medium and Hard tasks, which typically involve more intricate constraints and complex business logic. Overall, these results demonstrate the robustness of semantic gradients in handling complex real-world constraints.

4.5 Ablations and Directional Analysis

We evaluate the contributions of SGEvolve's core components and assess the effectiveness and directional consistency of semantic-gradient guidance.

Component ablations. We first conduct ablation studies on Circle Packing and Heilbronn (triangle). Averaging across three independent runs per task, we examine three settings: removing the semantic

gradient extraction operator ($\mathcal{E}_\theta$), removing crossover, and removing mutation (Table 4). Removing semantic gradient extraction reduces both mean and best performance on the two tasks, highlighting the contribution of feedback derived from evaluated parent–offspring transitions. Both Mutation-Only and Crossover-Only outperform the variant without semantic gradient extraction in mean performance on both tasks. Crossover-Only exhibits greater run-to-run variability than Mutation-Only, as reflected by its larger standard deviations.

Together, these component ablations support the complementary roles of mutation and crossover in combining local refinement with the recombination of elite candidates.

Directional guidance versus reflection. The extraction ablation removes both directional guidance and reflective information. To distinguish their contributions, we evaluate a matched Reflection-only variant across all five geometry tasks. This variant keeps the mutation and crossover operators, population management, backbone, and inference budget unchanged, replacing semantic-gradient extraction with free-form feedback on what worked, what failed, and how to improve. Full SGEvolve achieves better mean performance on all five tasks. For example, the mean Moving Sofa objective increases from 1.932 to 2.220, and the mean 7D Sphere Packing objective increases from 7.679 to 7.837. This matched comparison supports the contribution of structured directional feedback beyond free-form reflection. Detailed results are provided in Appendix B.3.

Directional consistency. We next examine whether the extracted directions are associated with fitness changes in subsequent individual mutations. We analyze $1{,}143$ mutation events across the five geometry tasks using the semantic rule attached to each parent before offspring generation. With a frozen external encoder e, we compute $A_t = \cos(e(g_t), e(s_o) - e(s_p))$ and define the directional consistency rate (DCR) as the fraction of events for which $A_t \Delta\widetilde{\mathcal{F}}_t > 0$, where $\Delta\widetilde{\mathcal{F}}_t$ is the offspring-minus-parent fitness difference with all objectives expressed in maximization form. The pooled DCR is 0.583, and the pooled odds ratio is 2.172 with a 95% confidence interval of $[1.674, 2.825]$. Three of the five tasks retain significant positive associations after Holm correction. This prospective association provides operational support for the positive-alignment condition in Proposition 1, with external embeddings serving as a proxy rather than a direct measurement of the internal latent gradient. Detailed results are provided in Appendix B.4.

5 Conclusions

In this paper, we presented SGEvolve, a novel framework bridging heuristic LLM-driven evolutionary search with principled zeroth-order optimization. To overcome the directionless nature of stochastic variations in discrete text spaces, we introduced the conceptual framework of semantic gradients. By viewing performance differences between parent and offspring through the lens of MAP estimation, SGEvolve decodes latent ascent directions into explicit textual guidance. This structural anchor heuristically justifies a positive expected fitness gain, enabling more directed search trajectories. Empirically, SGEvolve effectively breaks the "evolutionary stagnation" of unguided baselines, achieving consistent state-of-the-art performance across mathematical optimization, complex algorithm design, and highly constrained industrial problems. Furthermore, our ablations confirm that strategically coupling gradient-guided mutation and crossover optimally balances search stability with effective representation recombination.

Looking forward, the formalization of semantic gradients opens promising avenues, such as exploring higher-order analogies like "semantic momentum" or adaptive step-size control. Additionally, since the quality of gradient extraction intrinsically relies on the LLM's pre-trained prior, adapting this framework for completely out-of-distribution (OOD) novel discoveries remains an exciting challenge. Ultimately, grounding LLM-driven semantic search in mathematical optimization principles provides a scalable and reliable blueprint for building autonomous, self-improving AI agents.

Acknowledgments and Disclosure of Funding

This work was supported by Zhongguancun Academy (Grant No. c20250507). The authors declare no competing interests.

References

[1] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

[2] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VNckp7JEHn.

[3] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

[4] Nicolas Legrand, Kenneth Enevoldsen, Márton Kardos, and Kristoffer Nielbo. Improving reasoning at inference time via uncertainty minimisation, 2026. URL https://arxiv.org/abs/2603.07159.

[5] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, January 2025. URL https://doi.org/10.48550/arXiv.2501.12948.

[6] Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling llm inference-time compute with adaptive branching tree search. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 35448–35484. Curran Associates, Inc., 2025. doi: 10.52202/085713-1191. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/32dff2e27b0da1fb5f4209216a948544-Paper-Conference.pdf.

[7] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *NATURE*, 625(7995), JAN 18 2024. ISSN 0028-0836. doi: 10.1038/s41586-023-06924-6.

[8] Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Zhe Zhao, Xialiang Tong, Kun Mao, Zhichao Lu, Zhenkun Wang, Mingxuan Yuan, and Qingfu Zhang. A systematic survey on large language models for algorithm design. *ACM Comput. Surv.*, 58(8), February 2026. ISSN 0360-0300. doi: 10.1145/3787585. URL https://doi.org/10.1145/3787585.

[9] Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *CoRR*, abs/2506.13131, June 2025. URL https://doi.org/10.48550/arXiv.2506.13131.

[10] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

[11] Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=BwAkaxqiLB.

[12] Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lKEdGCoDNC.

[13] Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, Alex Dimakis, and Ion Stoica. Adaevolve: Adaptive llm driven zeroth-order optimization, 2026. URL https://arxiv.org/abs/2602.20133.

[14] Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, Hao Cheng, Pengcheng He, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Thetaevolve: Test-time learning on open problems, 2025. URL https://arxiv.org/abs/2511.23473.

[15] Mert Yüksekgönül, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time. *CoRR*, abs/2601.16175, January 2026. URL https://doi.org/10.48550/arXiv.2601.16175.

[16] Nian Ran, Zhongzheng Li, Yue Wang, Qingsong Ran, Xiaoyuan Zhang, Shikun Feng, Richard Allmendinger, and Xiaoguang Zhao. MCCE: A framework for multi-LLM collaborative search in discrete spaces with similarity-filtered preference learning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 103698–103716. PMLR, 2026. URL https://proceedings.mlr.press/v306/ran26a.html.

[17] Asankhaya Sharma. Openevolve: An open-source evolutionary coding agent. https://github.com/algorithmicsuperintelligence/openevolve, 2025. GitHub repository.

[18] ChentongChen, Mengyuan Zhong, Jialong Shi, Jianyong Sun, and Ye Fan. Hifo-prompt: Prompting with hindsight and foresight for LLM-based automatic heuristic design. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=imSLzfZ6av.

[19] Nian Ran, Yue Wang, Xiaoyuan Zhang, Zhongzheng Li, Qingsong Ran, Wenhao Li, and Richard Allmendinger. Exllm: Experience-enhanced llm optimization for molecular design and beyond, 2025. URL https://arxiv.org/abs/2502.12845.

[20] Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. *CoRR*, abs/2212.07677, 2022. URL https://doi.org/10.48550/arXiv.2212.07677.

[21] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT learn in-context? language models implicitly perform gradient descent as meta-optimizers. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023. URL https://openreview.net/forum?id=fzbHRjAd8U.

[22] Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=483IPGOHWL.

[23] Minghao Yan, Bo Peng, Benjamin Coleman, Ziqi Chen, Zhouhang Xie, Shuo Chen, Zhankui He, Noveen Sachdeva, Isabella Ye, Weili Wang, Chi Wang, Ed H. Chi, Fernando Pereira, Wang-Cheng Kang, Derek Zhiyuan Cheng, and Beidou Wang. Pacevolve: Enabling progress-aware consistent evolution, 2026. URL https://arxiv.org/abs/2601.10657.

[24] Jiachen Jiang, Tianyu Ding, and Zhihui Zhu. Deltaevolve: Accelerating scientific discovery through momentum-driven evolution, 2026. URL https://arxiv.org/abs/2602.02919.

[25] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=WRYhaSrThy.

[26] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text, 2024. URL https://arxiv.org/abs/2406.07496.

[27] Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. Unleashing the potential of large language models as prompt optimizers: Analogical analysis with gradient-based model optimizers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25264–25272, Apr. 2025. doi: 10.1609/aaai.v39i24.34713. URL https://ojs.aaai.org/index.php/AAAI/article/view/34713.

[28] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

[29] Reiko Tanese. Distributed genetic algorithms for function optimization. 1989. URL https://api.semanticscholar.org/CorpusID:60559318.

[30] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909, 2015. URL http://arxiv.org/abs/1504.04909.

[31] Google. Gemini 3 flash: frontier intelligence built for speed. https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/, 2025.

[32] Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale, 2025. URL https://arxiv.org/abs/2511.02864.

[33] Joseph L. Gerver. On moving a sofa around a corner. *Geometriae Dedicata*, 42:267–283, 1992. URL https://api.semanticscholar.org/CorpusID:119520847.

[34] Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. Alebench: A benchmark for long-horizon objective-driven algorithm engineering, 2025. URL https://arxiv.org/abs/2506.09050.

[35] Zhengyang Tang, Chenyu Huang, Xin Zheng, Shixi Hu, Zizhuo Wang, Dongdong Ge, and Benyou Wang. Orlm: Training large language models for optimization modeling. *arXiv preprint arXiv:2405.17743*, 2024.

A Proof of Proposition 1

Assumption (Positive Alignment). Since a single exploration step δ_z provides only directional information, we consider the expected behavior of semantic gradient extraction. The MAP alignment objective in Equation 5 motivates seeking a semantic direction that explains the observed fitness variation. At a fixed base state ϕ_{base} and search context, we assume that the extracted semantic gradient positively aligns with the true steepest-ascent direction in expectation. Formally, there exists a constant $\mu > 0$ such that:

$$\mathbb{E}\left[\langle \nabla f(\phi_{\text{base}}), \psi(\mathbf{g}^*)\rangle\right] \geq \mu \|\nabla f(\phi_{\text{base}})\|^2.$$

This condition gives a strictly positive expected directional inner product when $\nabla f(\phi_{\text{base}}) \neq \mathbf{0}$.

Proof. Let $Z_{new} = \phi(s_{new})$ be the continuous latent state of the offspring generated by our operators. According to the semantic gradient-guided generation distribution defined in Equation 6, Z_{new} can be decomposed into a deterministic semantic shift and a zero-mean stochastic sampling noise ξ:

$$Z_{\text{new}} = \phi_{\text{base}} + \eta\psi(\mathbf{g}^*) + \xi \tag{8}$$

where $\mathbb{E}[\xi] = \mathbf{0}$, and ξ is assumed to be independent of $\mathbf{g}^*$.. We denote the sampling variance introduced by the LLM decoding process as $V = \mathbb{E}[\|\xi\|^2] < \infty$, and assume $\mathbb{E}_{\mathbf{g}^*}[\|\psi(\mathbf{g}^*)\|^2] < \infty$.

By utilizing the L-smoothness property of the surrogate function f, the fitness of any generated individual satisfies:

$$f(Z_{\text{new}}) \geq f(\phi_{\text{base}}) + \langle \nabla f(\phi_{\text{base}}), Z_{\text{new}} - \phi_{\text{base}}\rangle - \frac{L}{2}\|Z_{\text{new}} - \phi_{\text{base}}\|^2 \tag{9}$$

We take the expectation with respect to the LLM's generation noise ξ, conditioned on a specific inferred rule $\mathbf{g}^*$. Using the zero-mean assumption and the variance decomposition, we have:

$$\mathbb{E}[f(Z_{\text{new}})] \geq f(\phi_{\text{base}}) + \eta\langle \nabla f(\phi_{\text{base}}), \psi(\mathbf{g}^*)\rangle - \frac{L}{2}\left(\eta^2\|\psi(\mathbf{g}^*)\|^2 + V\right) \tag{10}$$

Let $\Delta f = f(Z_{new}) - f(\phi_{base})$. Taking the total expectation over the inferred rule $\mathbf{g}^*$ and applying the assumption above, we obtain the expected fitness improvement:

$$\mathbb{E}[\Delta f] \geq \eta\mu\|\nabla f(\phi_{\text{base}})\|^2 - \frac{L\eta^2}{2}\mathbb{E}_{\mathbf{g}^*}\left[\|\psi(\mathbf{g}^*)\|^2\right] - \frac{L}{2}V \tag{11}$$

Table 5: Mathematical optimization benchmarks used in SGEvolve.

Problem	Objective
Circle packing $N = 26$	Arrange N non-overlapping circles within a unit square to **maximize the sum of their radii**.
7D sphere packing $N = 26$	Arrange N non-overlapping hyperspheres in a 7D hypercube to **maximize the sum of their radii**.
Heilbronn (triangle) $N = 11$	Place N points inside a unit-area equilateral triangle to **maximize the minimum area among all triangles formed by any three points**.
Moving sofa	Design a shape that can move through a right-angled corridor of width 1, **maximizing the area of the shape**.
Max-to-min ratios $N = 16$	Place N points in a 2D plane to **minimize the ratio between the maximum and minimum pairwise distances**.

Given that $\nabla f(\phi_{base}) \neq \mathbf{0}$, the first-order gain term $\mu\|\nabla f(\phi_{base})\|^2$ is strictly positive. Therefore, there exists an optimal step size η^* such that, as long as the LLM generation variance V is bounded by

$$V < \frac{2\eta\mu}{L}\|\nabla f(\phi_{\text{base}})\|^2 - \eta^2\mathbb{E}_{\mathbf{g}^*}\left[\|\psi(\mathbf{g}^*)\|^2\right] \tag{12}$$

we can theoretically guarantee a strictly positive expected fitness improvement, i.e., $\mathbb{E}[\Delta\mathcal{F}] > 0$.

Discussion. This derivation theoretically justifies the advantage of semantic gradient-guided operators over traditional randomized variations. In standard isotropic mutations (where $\mathbb{E}[\delta_{baseline}] = \mathbf{0}$), the expected first-order directional gain is exactly zero ($\mathbb{E}[\langle\nabla f, \delta_{baseline}\rangle] = 0$). Such methods must rely on subsequent population selection to filter out negative variations and secure an improvement. In contrast, under the positive-alignment assumption, semantic-gradient guidance contributes a positive first-order term $\eta\mu\|\nabla f(\phi_{\text{base}})\|^2$ to the lower bound on expected surrogate fitness improvement. When this term exceeds the curvature and sampling-noise penalties, as specified in Equation 12, the resulting lower bound is strictly positive, yielding positive expected surrogate fitness improvement prior to selection.

B Experimental Details

B.1 Hyperparameters and Budget Settings

As described in Section 4, we conduct three groups of experiments. Within each group, SGEvolve uses a fixed set of hyperparameters and computational budgets.

For **mathematical optimization**, we set the mutation probability to 0.45 and the crossover probability to 0.55. The number of islands is 2, with a population size of 20 per island. Population migration is performed every 25 iterations. The computational budget for each problem is limited to $5.

For **ALE-Bench LITE**, we set the mutation probability to 0.5 and the crossover probability to 0.5. The number of islands is 2, with a population size of 10 per island. Migration occurs every 20 iterations. The per-task computational budget is limited to $3.

For **IndustryOR**, we set the mutation probability to 0.45 and the crossover probability to 0.55. The number of islands is 1, with a population size of 10 per island. Migration is performed every 8 iterations. Due to the large number of tasks in this benchmark, the computational budget per task is limited to $2.

Here, the budget (denoted in $) represents a unified cost constraint that accounts for all LLM inference calls during the evolutionary process, ensuring fair comparison across methods.

B.2 Mathematical Optimization Benchmarks Details

Table 5 summarizes the configurations and optimization objectives of the mathematical optimization benchmarks. More detailed mathematical formulations can be found in Georgiev et al. [32], Gerver [33].

Table 6: Full SGEvolve and Reflection-only on five geometry tasks. Entries report mean ± standard deviation and best objective values. All displayed values are rounded to the reported precision.

	SGEvolve		Reflection-only	
Task	Mean ± std	Best	Mean ± std	Best
Circle Packing ↑	2.636 ± 0.000	2.636	2.622 ± 0.019	2.635
7D Sphere Packing ↑	7.837 ± 0.020	7.865	7.679 ± 0.047	7.733
Heilbronn Triangle ↑	0.0365 ± 0.0000	0.0365	0.0342 ± 0.0002	0.0346
Moving Sofa ↑	2.220 ± 0.001	2.221	1.932 ± 0.150	2.092
Max-to-Min Ratios ↓	3.590 ± 0.000	3.590	3.602 ± 0.006	3.598

B.3 Reflection-only Comparison

To isolate the contribution of directional feedback, we compare full SGEvolve with a Reflection-only variant on the five geometry tasks. Both variants use the same backbone, candidate initialization protocol, mutation and crossover operators, population management, evaluator, and inference-cost budget. Reflection-only replaces the extraction prompt with free-form feedback on what worked, what failed, and how to improve, without invoking semantic gradients, MAP estimation, or ascent directions. The resulting feedback is passed unchanged to the same variation operators.

Both variants use Gemini-3-Flash. Each run has a total LLM inference budget of $5, including feedback generation and other auxiliary LLM calls. The shared evolutionary settings follow Appendix B.1. The full prompts are provided in Appendix C.2.

Full SGEvolve achieves better mean performance on all five tasks (Table 6), supporting the contribution of structured directional feedback beyond free-form reflection under matched evolutionary settings.

B.4 Directional Consistency Analysis

Proposition 1 characterizes sufficient conditions under which semantic-gradient guidance yields positive expected improvement. To examine the empirical plausibility of its positive-alignment condition, we analyze directional consistency in actual parent–offspring transitions across all five geometry tasks.

For each mutation event t, we record the parent $s_{p,t}$, the offspring $s_{o,t}$, and the semantic gradient g_t already associated with the parent before offspring generation. Using a frozen external embedding encoder $e(\cdot)$, we compute

$$A_t = \cos\left(e(g_t), e(s_{o,t}) - e(s_{p,t})\right). \tag{13}$$

Since g_t exists before the current offspring is generated, the analysis is prospective. We express all objectives in maximization form, negating minimization objectives, and define

$$\Delta\widetilde{\mathcal{F}}_t = \widetilde{\mathcal{F}}(s_{o,t}) - \widetilde{\mathcal{F}}(s_{p,t}), \tag{14}$$

so that a positive value indicates an improvement.

We define the Directional Consistency Rate (DCR) as

$$\mathrm{DCR} = \frac{1}{N}\sum_{t=1}^{N} \mathbf{1}\left[A_t \Delta\widetilde{\mathcal{F}}_t > 0\right]. \tag{15}$$

A transition is directionally consistent if it follows the semantic direction and improves fitness, or opposes that direction and degrades fitness. We use sign consistency because fitness differences can become small near convergence while their improvement or degradation signs remain informative.

For each task, we construct the following sign contingency table:

	$\Delta\widetilde{\mathcal{F}}_t > 0$	$\Delta\widetilde{\mathcal{F}}_t < 0$
$A_t > 0$	n_{++}	n_{+-}
$A_t < 0$	n_{-+}	n_{--}

. (16)

Table 7: Sign contingency counts and directional consistency across the five geometry tasks. Each event is a mutation evaluated using the semantic gradient associated with its parent before offspring generation.

Task	Events	n_{++}	n_{+-}	n_{-+}	n_{--}	DCR
Circle Packing	226	103	69	27	27	0.575
7D Sphere Packing	241	102	69	29	41	0.593
Heilbronn Triangle	217	71	71	23	52	0.567
Moving Sofa	184	57	56	10	61	0.641
Max-to-Min Ratios	275	96	74	49	56	0.553
Pooled	1143	429	339	138	237	0.583

Table 8: Task-level association between projected semantic directions and subsequent fitness changes. The final column reports one-sided exact-test p-values adjusted using Holm correction over five tasks.

Task	DCR	OR	Exact 95% CI	Adjusted p
Circle Packing	0.575	1.490	$[0.769, 2.890]$	0.1449
7D Sphere Packing	0.593	2.083	$[1.144, 3.835]$	0.0224
Heilbronn Triangle	0.567	2.252	$[1.207, 4.291]$	0.0181
Moving Sofa	0.641	6.148	$[2.775, 14.847]$	1.37×10^{-6}
Max-to-Min Ratios	0.553	1.480	$[0.883, 2.490]$	0.1449

For the events represented in these tables, $N = n_{++} + n_{+-} + n_{-+} + n_{--}$ and

$$\mathrm{DCR} = \frac{n_{++} + n_{--}}{n_{++} + n_{+-} + n_{-+} + n_{--}}. \tag{17}$$

Table 7 reports the complete counts. Across the five tasks, 666 of the 1,143 analyzed transitions are directionally consistent, giving a pooled DCR of 0.583.

A DCR above 0.5 is descriptive and does not by itself establish statistical significance. We therefore test positive association between the two signs using the hypotheses

$$H_0 : \mathrm{OR} \leq 1, \qquad H_1 : \mathrm{OR} > 1. \tag{18}$$

The odds ratio compares the odds of fitness improvement when the projected offspring step follows the semantic direction ($A_t > 0$) with those when it opposes the direction ($A_t < 0$). We apply one-sided exact tests to the task-level contingency tables, use Holm correction for the five task-level comparisons, and report odds-ratio estimates with exact 95% confidence intervals in Table 8.

All five tasks have DCR values above 0.5 and odds-ratio estimates above 1. After Holm correction, positive associations remain significant on 7D Sphere Packing, Heilbronn Triangle, and Moving Sofa. Circle Packing and Max-to-Min Ratios also have positive point estimates, but their task-level evidence is insufficient to reject the null hypothesis after correction.

Pooling all 1,143 mutation events gives

$$\mathrm{OR}_{\mathrm{pooled}} = 2.172, \qquad 95\%\,\mathrm{CI} = [1.674, 2.825], \qquad p = 9.05 \times 10^{-10}. \tag{19}$$

Under this embedding-based proxy, direction-following steps have approximately 2.17 times the odds of fitness improvement of direction-opposing steps. To account for differences across tasks, we also perform a task-stratified Mantel–Haenszel analysis, obtaining

$$\mathrm{OR}_{\mathrm{MH}} = 2.112. \tag{20}$$

The similar pooled and stratified estimates indicate that the positive association persists after accounting for task strata.

These results provide operational support for the empirical plausibility of the positive-alignment condition in Proposition 1: candidate-specific semantic directions are positively associated with subsequent fitness changes under the external representation. This evidence concerns embedding-based directional consistency and does not directly recover the backbone LLM's unobservable internal gradient. The proposition therefore remains a conditional sufficient-condition result, while the analysis establishes the corresponding observable directional association in actual search transitions.

Table 9: SGEvolve with Gemini-3-Flash and DeepSeek-V4-Flash on five geometry tasks. Entries report the mean $\pm$ standard deviation and the best objective value across independent runs. All displayed values are rounded to the reported precision.

	Gemini-3-Flash		DeepSeek-V4-Flash	
Task	Mean $\pm$ std	Best	Mean $\pm$ std	Best
Circle Packing $\uparrow$	2.636 ± 0.000	2.636	2.636 ± 0.000	2.636
7D Sphere Packing $\uparrow$	7.837 ± 0.020	7.865	7.839 ± 0.015	7.860
Heilbronn Triangle $\uparrow$	0.0365 ± 0.0000	0.0365	0.0365 ± 0.0000	0.0365
Moving Sofa $\uparrow$	2.220 ± 0.001	2.221	2.221 ± 0.000	2.221
Max-to-Min Ratios $\downarrow$	3.590 ± 0.000	3.590	3.592 ± 0.002	3.590

B.5 Evaluation with an Open-Weight Backbone

To examine whether SGEvolve is applicable beyond Gemini-3-Flash, we evaluate it on the five geometry tasks using the open-weight DeepSeek-V4-Flash backbone. We keep the task definitions, evaluators, evolutionary operators, population-management strategy, and inference budget unchanged. The shared evolutionary settings follow Appendix B.1, with a total LLM inference budget of $5 per task per run.

Table 9 shows comparable mean and best performance across the five geometry tasks. These results support the applicability of SGEvolve across the two tested model families.

C Implementation Details

C.1 SGEvolve Prompts

We report the prompt templates used by SGEvolve.

Prompt 1 (*Semantic Gradient Extraction operation*).

```
# [SYSTEM] Evolutionary Semantic Gradient Extraction & Analysis
You perform a deep diagnostic analysis of a candidate implementation to distill actionable
intelligence for subsequent iterations.

## [1] Role Description
You are an Evolutionary Architect. Your task is to perform an analysis of a "Child"
candidate by comparing it with its "Parent" ancestor. You must determine why certain
structural changes led to specific performance outcomes and provide a blueprint for the
next generation.
Focus on providing actionable insights that can guide future improvements and refinements
of the child.

## [2] Problem Description
{PROBLEM_DESCRIPTION}

### Objective(s)
{OBJECTIVE}

### Constraints
{CONSTRAINTS}

## [3] Expert Insights
{EXPERT_INSIGHTS}

## [4] Lineage Context (Parent)
{PARENTAL_CONTEXT}

## [5] Child Candidate (Semantic Gradient Target)
### Candidate Evaluation Results
{CHILD_FITNESS}
```

````
Evaluation details:
{CHILD_EVAL_DETAILS}

Error logs & execution trace:
{CHILD_EVAL_LOGS}

### Candidate's Implementation
```{LANGUAGE}
{CHILD_EVOL_ENTITY}
```

## [6] Semantic Gradient Extraction Instructions
Please analyze the generated child's implementation in the context of its parent(s) and
provide the following information as components of the semantic gradient:
- "strength": Focus on the 'Evolutionary Gains'. What specific structural or logical
improvements does this child possess over its parent?
- "weakness": Identify 'Evolutionary Regressions' or 'Bottlenecks' if the child has. Where
does the child fail or underperform?
- "evolving_direction": Define the 'Evolutionary Trajectory' for the next generation. Based
on this child's performance, what specific strategic shift should be taken? (These could
differ from expert insights).
Do not mention the parents or any parent-specific information in your response; just
objectively describe the descendant itself.

# [7] Output Requirement
Output the results in the following JSON format:
```json
{OUTPUT_EXAMPLE}
```
````

Prompt 2 (*Gradient-Guided Mutation operation*).

````
# [SYSTEM] Direction-Guided Mutation Operator
You perform minimal, local code mutations that move the parent implementation strictly
along the given evolutionary direction.

## [1] Role Description
{ROLE_DESCRIPTION}

## [2] Problem Description
{PROBLEM_DESCRIPTION}

### Objective(s)
{OBJECTIVE}

### Constraints
{CONSTRAINTS}

## [3] Expert Insights
{EXPERT_INSIGHTS}

## [4] Evaluation and Reflection Feedback
### Evaluation Results
{FITNESS}

Evaluation details:
{EVAL_DETAILS}

Error logs & execution trace:
{EVAL_LOGS}

## [5] Content to Mutate
```{LANGUAGE}
{EVOL_ENTITY}
```

## [6] Evolutionary Direction
````

```
{EVOL_DIRECTION}

## [7] Mutation Task Instructions
1. Cross-reference the 'Source Code' with 'Evaluation details' and 'Error Logs'.
2. Apply the 'Expert Insights' and follow the 'Evolutionary Direction' to perform a
targeted mutation.
3. Your goal is to produce a "Descendant" that retains the core logic but surpasses the
parent in performance or correctness.

## [8] Output Format
Provide code in Markdown blocks only, without any other text.
Do not mention the parents or any parent-specific information in your response; if
descriptive text is required, objectively describe only the descendant itself.
```

Prompt 3 (*Gradient-Guided Crossover operation*).

```
# Direction-Guided Crossover Operator
You perform targeted recombination of two parent implementations to produce one descendant
that combine parents' strengths while mitigating their weaknesses.

## [1] Role Description
{ROLE_DESCRIPTION}

## [2] Problem Description
{PROBLEM_DESCRIPTION}

### Objective(s)
{OBJECTIVE}

### Constraints
{CONSTRAINTS}

## [3] Expert Insights
{EXPERT_INSIGHTS}

## [4] Evaluation and Reflection Feedback
### Parent A Evaluation Results
{PARENT_A_FITNESS}

Evaluation details:
{PARENT_A_EVAL_DETAILS}

Evaluation error logs:
{PARENT_A_EVAL_LOGS}

### Parent B Evaluation Results
{PARENT_B_FITNESS}

Evaluation details:
{PARENT_B_EVAL_DETAILS}

Error logs & execution trace:
{PARENT_B_EVAL_LOGS}

## [5] Content to Crossover
### Parent A
```{LANGUAGE}
{PARENT_A_EVOL_ENTITY}
```

Parent B
```{LANGUAGE}
{PARENT_B_EVOL_ENTITY}
```

[6] Parent Strengths & Weaknesses
Parent A -- Strengths
```

```
{PARENT_A_STRENGTH}

### Parent A -- Weaknesses
{PARENT_A_WEAKNESS}

### Parent B -- Strengths
{PARENT_B_STRENGTH}

### Parent B -- Weaknesses
{PARENT_B_WEAKNESS}

## [7] Crossover Task Instructions
1. Review both parents' logic, metrics and evaluation results; identify complementary
strengths.
2. Focus on high-impact blocks (function/class/block) where fusion or recombination yields
the most significant performance gain.
3. Produce **one** best Descendant that: combines strengths and reduces weaknesses, and
keeps edits minimal.
4. Never return either parent verbatim or as the final output. Always produce a new,
meaningfully evolved variant through fusion or refactoring.
5. Prioritize substantive improvements in algorithmic quality, efficiency, and robustness
without adding unnecessary dependencies. Focus on real optimization, not superficial edits.

## [8] Output Format
Return ONLY the requested code block. Do not include any conversational filler,
explanations, or text outside the Markdown code fences.
Do not mention the parents or any parent-specific information in your response; if
descriptive text is required, objectively describe only the descendant itself.
```

C.2 Reflection-only Prompt

```
# [SYSTEM] Pure Evolutionary Reflection
You reflect on a candidate implementation to produce concise, actionable feedback for
subsequent iterations.

## [1] Role Description
You are an Evolutionary Architect. Analyze the candidate below using its implementation,
evaluation results, and runtime behavior.

## [2] Problem Description
{PROBLEM_DESCRIPTION}

### Objective(s)
{OBJECTIVE}

### Constraints
{CONSTRAINTS}

## [3] Expert Insights
{EXPERT_INSIGHTS}

## [4] Lineage Context (Parent)
{PARENTAL_CONTEXT}

## [5] Candidate for Reflection
### Candidate Evaluation Results
{CHILD_FITNESS}

Evaluation details:
{CHILD_EVAL_DETAILS}

Error logs & execution trace:
{CHILD_EVAL_LOGS}

### Candidate's Implementation
‘‘‘{LANGUAGE}
{CHILD_EVOL_ENTITY}
```

```
```

[5] Reflection Instructions
Reflect on what worked, what did not work, and suggest how the candidate may be improved.
Store all feedback in the single "reflections" field.

[6] Output Requirement
Output the results in the following JSON format:
```json
{OUTPUT_EXAMPLE}
```
```

### C.3 An Example of Semantic Gradient Extraction

We illustrate one mutation realization on the $N{=}26$ circle-packing benchmark: the parent program, the offspring after semantic-gradient-guided mutation, and the extracted semantic gradient.

**Parent.** Fitness: 2.629760

```python
import numpy as np
from scipy.optimize import minimize

def search_algorithm(num_circles=26):
    n = num_circles

    def objective(x):
        return -np.sum(x[2*n:])

    def constraints(x):
        c = x[:2*n].reshape((n, 2))
        r = x[2*n:]
        # Boundary constraints: c - r >= 0 and 1 - c - r >= 0
        b_cons = np.concatenate([c - r[:, np.newaxis], 1.0 - c - r[:, np.newaxis]], axis=
    None)
        # Vectorized non-overlap constraints: dist(c_i, c_j) - (r_i + r_j) >= 0
        o_cons = []
        for i in range(n):
            diffs = c[i+1:] - c[i]
            dists = np.sqrt(np.sum(diffs**2, axis=1) + 1e-12)
            o_cons.append(dists - (r[i] + r[i+1:]))
        return np.concatenate([b_cons, *o_cons])

    def bubble_growth(c_init, iterations=100):
        c = c_init.copy()
        r = np.full(n, 0.05)
        for _ in range(iterations):
            for i in range(n):
                diffs = c[i] - c
                dists = np.linalg.norm(diffs, axis=1) + 1e-12
                overlap_mask = (dists < (r[i] + r)) & (np.arange(n) != i)
                if np.any(overlap_mask):
                    push = (r[i] + r[overlap_mask]) - dists[overlap_mask]
                    dirs = diffs[overlap_mask] / dists[overlap_mask][:, None]
                    c[i] += np.sum(dirs * push[:, None], axis=0) * 0.5
                # Keep centers in bounds considering radius
                c[i] = np.clip(c[i], r[i], 1.0 - r[i])
            r += 0.0005
        return np.concatenate([c.flatten(), r])

    def get_seeds():
        seeds = []
        # 1. 5x5 Grid + 1 Greedy Circle (Parent A/B hybrid)
        g = np.linspace(0.12, 0.88, 5)
        xv, yv = np.meshgrid(g, g)
        grid_pts = np.stack([xv.ravel(), yv.ravel()], axis=1)
        # Search for best 26th spot
```
```

```
    samples = np.random.rand(500, 2)
    d_walls = np.min(np.concatenate([samples, 1.0 - samples], axis=1), axis=1)
    d_grid = np.min(np.linalg.norm(samples[:, None, :] - grid_pts[None, :, :], axis=2) -
 0.09, axis=1)
    best_26 = samples[np.argmax(np.minimum(d_walls, d_grid))]
    seeds.append(np.vstack([grid_pts, best_26]))

    # 2. Phyllotactic (Spiral) Topology
    phi = np.pi * (3.0 - np.sqrt(5.0))
    idx = np.arange(n) + 0.5
    rads_pos = 0.44 * np.sqrt(idx / n)
    thetas = idx * phi
    seeds.append(0.5 + np.column_stack([rads_pos * np.cos(thetas), rads_pos * np.sin(
thetas)]))

    # 3. Pure Greedy Discovery
    c_greedy, r_greedy = [], []
    for i in range(n):
        s = np.random.rand(600, 2)
        dw = np.min(np.minimum(s, 1-s), axis=1)
        if i > 0:
            dc = np.min(np.linalg.norm(s[:, None, :] - np.array(c_greedy)[None, :, :],
axis=2) - np.array(r_greedy), axis=1)
            score = np.minimum(dw, dc)
        else: score = dw
        idx_best = np.argmax(score)
        c_greedy.append(s[idx_best]); r_greedy.append(score[idx_best])
    seeds.append(np.array(c_greedy))

    return seeds

# --- Phase 1: Exploration and Ranking ---
best_candidate_x = None
best_score = np.inf

for seed_c in get_seeds():
    x0 = bubble_growth(seed_c, iterations=120)
    # Quick SLSQP to rank seeds
    res = minimize(
        objective, x0, method='SLSQP',
        constraints={'type': 'ineq', 'fun': constraints},
        bounds=[(0, 1)]*(2*n) + [(0.0, 0.5)]*n,
        options={'maxiter': 100, 'ftol': 1e-6}
    )
    if res.fun < best_score:
        best_score = res.fun
        best_candidate_x = res.x

# --- Phase 2: High-Precision Polish ---
# Final optimization with tightened convergence criteria and higher iteration limit
final_res = minimize(
    objective, best_candidate_x, method='SLSQP',
    constraints={'type': 'ineq', 'fun': constraints},
    bounds=[(0, 1)]*(2*n) + [(0.0, 0.5)]*n,
    options={'maxiter': 500, 'ftol': 1e-12}
)

# If final polish didn't improve or failed, fall back to best candidate
if final_res.success and final_res.fun < best_score:
    result_x = final_res.x
else:
    result_x = best_candidate_x

return result_x[:2*n].reshape((n, 2)), result_x[2*n:]
```

Child. Fitness: 2.635983

```python
import numpy as np
from scipy.optimize import minimize

def search_algorithm(num_circles=26):
    n = num_circles

    def objective(x):
        # Minimize the negative sum of radii to maximize the total sum
        return -np.sum(x[2*n:])

    def constraints(x):
        c = x[:2*n].reshape((n, 2))
        r = x[2*n:]
        # Boundary constraints: center - radius >= 0 AND center + radius <= 1
        b_cons = np.concatenate([c - r[:, np.newaxis], 1.0 - (c + r[:, np.newaxis])], axis=
    None)
        # Non-overlap constraints: distance between centers - sum of radii >= 0
        o_cons = []
        for i in range(n):
            diffs = c[i+1:] - c[i]
            dists = np.sqrt(np.sum(diffs**2, axis=1) + 1e-15)
            o_cons.append(dists - (r[i] + r[i+1:]))
        return np.concatenate([b_cons, *o_cons])

    def adaptive_bubble_growth(c_init, iterations=150):
        c = c_init.copy()
        # Start with a conservative radius estimate
        r = np.full(n, 0.085)
        for step in range(iterations):
            for i in range(n):
                # Repulsion from other circles
                diffs = c[i] - c
                dists = np.linalg.norm(diffs, axis=1) + 1e-12
                overlap = (r[i] + r) - dists
                mask = (overlap > 0) & (np.arange(n) != i)
                if np.any(mask):
                    push_dirs = diffs[mask] / dists[mask][:, None]
                    c[i] += np.sum(push_dirs * overlap[mask][:, None], axis=0) * 0.45

                # Containment within bounds
                c[i] = np.clip(c[i], r[i], 1.0 - r[i])

            # Incremental growth
            r += 0.0002
            # Occasional jitter to break symmetry traps
            if step % 25 == 0:
                c += np.random.normal(0, 0.001, c.shape)
        return np.concatenate([c.flatten(), r])

    def get_seeds():
        seeds = []
        # 1. High-Density Greedy Discovery
        for _ in range(2):
            c_greedy, r_greedy = [], []
            for i in range(n):
                # 2000 candidate points per circle to find optimal local placement
                s = np.random.rand(2000, 2)
                dw = np.min(np.minimum(s, 1-s), axis=1)
                if i > 0:
                    dists = np.linalg.norm(s[:, None, :] - np.array(c_greedy)[None, :, :],
    axis=2)
                    dc = np.min(dists - np.array(r_greedy), axis=1)
                    score = np.minimum(dw, dc)
                else:
                    score = dw
                idx_best = np.argmax(score)
                c_greedy.append(s[idx_best])
                r_greedy.append(max(0.01, score[idx_best]))
```

```
            seeds.append(np.array(c_greedy))

        # 2. Distorted 5x5 Grid (n=25) + 1 Randomized Center
        g = np.linspace(0.1, 0.9, 5)
        xv, yv = np.meshgrid(g, g)
        grid = np.stack([xv.ravel(), yv.ravel()], axis=1)
        # Introduce slight defects to the grid to assist non-uniform optimization
        grid += np.random.uniform(-0.015, 0.015, grid.shape)
        extra = np.random.rand(1, 2)
        seeds.append(np.vstack([grid, extra]))

        return seeds

    # --- Phase 1: Exploration ---
    best_x = None
    best_val = np.inf

    for seed_c in get_seeds():
        x0 = adaptive_bubble_growth(seed_c)
        res = minimize(
            objective, x0, method='SLSQP',
            constraints={'type': 'ineq', 'fun': constraints},
            bounds=[(0, 1)]*(2*n) + [(0.0, 0.5)]*n,
            options={'maxiter': 150, 'ftol': 1e-7}
        )
        if res.fun < best_val:
            best_val = res.fun
            best_x = res.x

    # --- Phase 2: Refined Jittered Polish ---
    # Attempt to escape local minima by perturbing the best found topology
    current_best_x = best_x
    for attempt in range(2):
        x_shake = current_best_x.copy()
        if attempt > 0:
            x_shake[:2*n] += np.random.normal(0, 0.0005, 2*n)

        final_res = minimize(
            objective, x_shake, method='SLSQP',
            constraints={'type': 'ineq', 'fun': constraints},
            bounds=[(0, 1)]*(2*n) + [(0.0, 0.5)]*n,
            options={'maxiter': 500, 'ftol': 1e-12}
        )

        if final_res.success and final_res.fun < best_val:
            best_val = final_res.fun
            current_best_x = final_res.x

    return current_best_x[:2*n].reshape((n, 2)), current_best_x[2*n:]
```

Semantic gradient (extracted signal).

Strengths.

- *Combines multiple seeding strategies including a 5x5 grid plus a greedy search for the 26th circle, phyllotactic spirals, and pure greedy discovery.*
- *Introduces a multi-phase optimization workflow: seed generation, physics-based pre-conditioning (bubble growth), rapid ranking of candidates, and a final high-precision polish.*
- *The use of a ranking phase (Phase 1) efficiently filters out less promising topologies before committing computational resources to high-precision optimization.*
- *Good use of vectorized constraints for the SLSQP solver, enhancing the gradient calculation efficiency.*

Weaknesses.

- *Despite the increased complexity, the result (2.62976) underperforms compared to Parent 1 (2.63429), suggesting that the bubble-growth phase might be disrupting high-quality initial configurations.*
- *The bubble growth starts with a fixed radius (0.05) for all circles, which may cause excessive overlapping forces in dense areas, leading to suboptimal 'jams' before the solver takes over.*
- *The ranking phase uses a fixed iteration limit (100), which might prematurely discard a topology that requires more steps to resolve overlaps but ultimately yields a higher sum of radii.*
- *The search for the 'best 26th spot' in the grid seed uses a fixed radius (0.09) in its distance calculation, which is arbitrary and may not reflect the actual space available.*

Evolving direction.

- *Incorporate Basin Hopping or Simulated Annealing: The current plateau suggests the SLSQP solver is getting trapped in local optima; a global optimization wrapper could help escape these.*
- *Adaptive Bubble Growth: Instead of a fixed radius increment, scale the growth based on the current packing density or local free space to prevent premature jamming.*
- *Symmetry-Specific Seeding: For n=26, research suggests the optimal packing involves a distorted square lattice; seeding with slight 'defects' or row-shifts in the 5x5 grid might break the symmetry trap.*
- *Constraint Softening: Start the optimization with slightly relaxed non-overlap constraints to allow circles to 'tunnel' through each other to reach better global configurations before hardening the constraints for the final polish.*

D Broader Impact

This work proposes SGEvolve, a framework to improve the efficiency of LLM-driven optimization. Positive impacts include accelerating scientific discovery and industrial design. Potential negative impacts include the risk of automating the generation of harmful code or dual-use algorithms, though this is a general risk associated with all LLM-based tools and not specific to our search framework.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly outline our three core contributions: (1) the formulation of semantic gradients, (2) the SGEvolve framework (including extraction, crossover, and mutation operators), and (3) empirical validation across diverse domains. These claims are consistently supported by the methodology and experimental results presented in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in the Conclusions section. Specifically, we acknowledge that the current framework: (1) relies on zero-order approximations rather than higher-order optimization, (2) is constrained by the LLM's pre-trained priors for MAP estimation, and (3) faces challenges in out-of-distribution (OOD) scenarios for completely novel discoveries.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper includes a formal proposition (Proposition 1) regarding the expected fitness improvement in Section 3.2. All necessary assumptions, such as the L-smoothness of the surrogate function and bounded sampling variance, are explicitly stated, and the complete proof is provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the backbone model (Gemini-3-Flash) and the benchmarks used (ALE-Bench, IndustryOR). It also details the experimental setup, computational budgets, and evaluation metrics, with specific hyperparameters and budget configurations provided in Appendix B.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The SGEvolve code is publicly available at the repository linked in the abstract, with reproduction instructions in the README. The benchmarks used are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 and Appendix B specify the backbone model, task benchmarks, and detailed hyperparameter configurations. As the framework performs evolutionary search rather than gradient-based training, these details provide the necessary context for understanding the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the Mathematical Optimization tasks (Section 4.2) and Ablation studies (Section 4.5), we report the mean performance averaged over 3 independent runs to account for the stochasticity of the LLM generation and the evolutionary search process. We discuss the statistical stability and variance in the text, and the corresponding tables report the standard deviation across these independent runs. For ALE-Bench and IndustryOR, we report the aggregate scores and success rates following the standard evaluation protocols of these benchmarks, bounded by the high computational cost of LLM inference.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The primary compute resource used is the Gemini-3-Flash model backbone via API calls. Since the framework relies on LLM inference rather than local training, specialized hardware like high-end GPUs is not required for reproduction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Our research develops a general-purpose optimization framework and evaluates it on standard public benchmarks. It does not involve human subjects, sensitive personal data, or any applications that violate the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the potential positive and negative societal impacts of our work in the Broader Impact section in the Appendix.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper introduces an algorithmic framework that utilizes existing commercial LLM APIs and standard, publicly available optimization benchmarks. It does not release new foundation models, high-risk datasets, or any assets that require specific safeguards against misuse or dual-use.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper properly credits the creators of all used assets, including the benchmarks (ALE-Bench, IndustryOR), the Gemini backbone model, and the baseline frameworks (ReEvo, AdaEvolve, etc.), through citations. These assets are utilized in accordance with their standard research purposes.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the SGEvolve code with a README describing the setup and usage in the repository linked in the abstract.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the central component of the proposed SGEvolve framework. Section 3 describes in detail how the LLM is used to extract semantic gradients and perform guided evolutionary operations.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.