FAILURE-DRIVEN WORKFLOW REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing LLM-based workflows is typically formulated as a global search, where candidate workflows are evaluated based on a scalar metric. This paradigm, however, suffers from a critical flaw: information collapse. By reducing rich, multi-step execution traces to simple success/failure signals, existing methods are rendered blind to the underlying structure of failures, fundamentally preventing them from modeling the workflow's failure distribution. We reconceptualize this challenge as a distributional problem. We propose a new paradigm where the optimization goal is not to maximize a scalar score, but to directly minimize a workflow's **Expected Failure Mass**, i.e., the integral of its failure probability density function defined over a high-dimensional **Failure Signature Space** ($\mathcal{F}$). This distributional lens allows us to move from inefficient, zero-order optimization to a principled, gradient-like descent on the failure landscape itself. We introduce **CE-Graph**, a framework that operationalizes this paradigm through a novel, failure-driven refinement process. CE-Graph approximates the failure distribution from a pool of counterexamples, identifies its densest regions as recurring failure modes, and applies targeted, operator-constrained graph edits via a **Propose-and-Verify** mechanism to greedily reduce the failure mass. On math, code, and QA benchmarks, our CE-Graph achieves higher robustness at a significantly lower cost than strong baselines. This suggests that a system's reliability emerges not from avoiding failures, but from systematically learning and reshaping the geometric structure of its failure distributions.

1 INTRODUCTION

Large Language Models (LLMs) (OpenAI et al., 2024; Vaswani et al., 2023; Zhao et al., 2025; Minaee et al., 2025) are increasingly used to power agentic workflows, where tasks are decomposed into multiple steps involving tool calls, logical control flows, and verification routines (Zhang et al., 2025c;a;b; Jeyakumar et al., 2024). Such workflows enable LLMs to handle long-horizon reasoning and complex problem-solving. The central challenge is how to optimize these workflows: given a dataset D, how can we construct a workflow W that achieves the highest reliability?

This challenge is fundamentally distinct from traditional program synthesis (Yu et al., 2025b; Sapkota et al., 2026). Unlike programs in a formal language, LLM-based workflows operate in a vast, unstructured space defined by natural language prompts and arbitrary tool interactions. Furthermore, their failures are not deterministic bugs but are often **stochastic and semantic** in nature, stemming from subtle flaws in the model's reasoning process (Steenhoek et al., 2025; Song et al., 2025; Huang et al., 2025). Consequently, established methods designed for formal program spaces are often ineffective, necessitating a new optimization paradigm.

A common formulation treats workflow optimization as a global program search (Zhang et al., 2025b;a; Li et al., 2024; Yu et al., 2025a). The objective is to find a configuration W^* that maximizes a global, sample-based evaluation metric $G(W, D)$: $W^* = \arg\max_{W \in S} G(W, D)$. Recent systems exemplify this view by employing advanced search strategies such as Monte Carlo Tree Search (MCTS) (Coulom, 2006; Browne et al., 2012; undefined-wiechowski et al., 2022). While effective, this line of work inherits a core limitation rooted in its objective function, i.e., the reliance on **information collapse**. The rich, multi-step trace of each failure is compressed into a single binary signal, preventing the optimizer from modeling the **underlying failure distribution**, which is key to efficient refinement.

We argue that these failures are not random noise but are, in fact, samples from a workflow-specific failure distribution (Pan et al., 2025; Zhang et al., 2025d). This insight motivates a paradigm shift. We re-conceptualize workflow optimization not as maximizing a scalar score, but as a process of progressively reshaping this underlying failure distribution. We formalize this by introducing the **Failure Signature Space** $\mathcal{F}$ and casting the optimization goal as the minimization of the **Expected Failure Mass** $M(W)$: $W^* = \arg\min_{W \in S} M(W)$ where $M(W) = \int_{\mathcal{F}} p(\mathbf{s} \mid W)\, d\mathbf{s}$.This new formulation, which aims to minimize the integral of a failure density function, marks a fundamental departure from the conventional paradigm of maximizing a sample-based performance metric. Instead of treating the workflow as a black box, it provides a white-box and distributional lens. This allows us to move from inefficient, zero-order optimization to a principled, gradient-like descent directly on the landscape of failure

modes. To operationalize this principle, we propose CE-Graph (Counterexample-Guided Workflow Optimization), a framework that realizes this failure-mass-reduction strategy. It operates through three key stages. First, it maintains a counterexample pool and systematically diagnoses the densest regions of the failure distribution by clustering their **semantic signatures**, thereby approximating the "semantic gradient". Second, it replaces heuristic fixes with a principled **Propose-and-Verify** mechanism, which generates and empirically validates targeted graph edits designed to reduce the mass of these failure modes. Third, our CE-Graph incorporates a convergence-aware stopping rule to halt optimization once the failure distribution has stabilized.

We evaluate our CE-Graph on math, code, and QA benchmarks, showing that it achieves higher success rates at significantly lower optimization cost compared to strong workflow search baselines. Beyond these empirical gains, our work highlights a fundamental principle for building robust agentic systems: true reliability emerges not from attempting to avoid failures, but from systematically understanding and resolving their underlying distributional structure. Conceptually, CE-Graph establishes a **self-referential optimization paradigm**, employing language models to analyze and refine the failure distributions of the very LLM-based systems they compose.

2 A New Optimization Paradigm for LLM Workflows

In this section, we develop the theoretical foundation for our CE-Graph by proposing a new paradigm for LLM workflow optimization. We begin by introducing the concept of a **Failure Signature Space** and define the optimization problem as minimizing the *expected failure mass* within this space (Ribeiro et al., 2020). This distributional perspective addresses the limitations of traditional scalar-based metrics by preserving the rich structure of failures. We then formalize the dominant "global search" paradigm and demonstrate its mathematical limitations in solving this objective, particularly its blindness to failure distributions (Zhang et al., 2025b). This motivates our new paradigm, "failure-driven refinement", which we articulate as a principled, iterative approach to minimizing this failure mass, thereby setting a rigorous stage for the algorithmic framework in Section 3. Our analysis reveals that true robustness in LLM workflows emerges from reshaping failure landscapes, rather than merely aggregating success rates.

2.1 Problem Formulation in the Failure Signature Space

The raw execution traces of LLM workflows are complex, high-dimensional, and unstructured, making direct comparison and analysis difficult. For instance, in a math reasoning task like GSM8K (Cobbe et al., 2021), a failure trace might involve multiple steps of erroneous calculations, tool calls, and reasoning flaws, which are hard to quantify without a unified representation. To enable systematic optimization, we map failures into a shared, structured vector space. Let $\mathcal{F}$ be this high-dimensional **Failure Signature Space**, where each point $\mathbf{s} \in \mathcal{F}$ represents a unique, structured signature of a potential workflow failure. Any given workflow W induces a probability density function $p(\mathbf{s} \mid W)$ over this space. This density function serves as a topographical map of the workflow's weaknesses: for a perfect workflow, the map is flat with $p(\mathbf{s} \mid W) = 0$; for a flawed workflow, "mountains" of high density emerge in regions corresponding to its systematic errors, such as recurring arithmetic mistakes in math problems. By modeling failures distributionally, we capture not just the frequency but the geometric clustering of errors, enabling targeted repairs. The ultimate goal of optimization is to find a workflow W^* that flattens this landscape, minimizing the total probability mass of failure. We formalize this as minimizing the **Expected Failure Mass** $M(W)$:

$$W^* = \arg\min_{W \in S} M(W) \quad \text{where} \quad M(W) = \int_{\mathcal{F}} p(\mathbf{s} \mid W)\, d\mathbf{s}. \tag{1}$$

This formulation provides a principled theoretical lens for analyzing different optimization paradigms, shifting the focus from black-box evaluation to white-box distributional refinement. This objective relies on several assumptions: (i) failures are independent and identically distributed (i.i.d.) samples from $p(\mathbf{s} \mid W)$, (ii) $\mathcal{F}$ captures all relevant semantic and structural aspects of failures via an appropriate embedding, and (iii) the density $p(\mathbf{s} \mid W)$ is estimable from finite counterexamples. Limitations include the curse of dimensionality in $\mathcal{F}$ and the non-convexity of $M(W)$, which may cause local minima. Nevertheless, this lens still enables gradient-like optimization, as explored next.

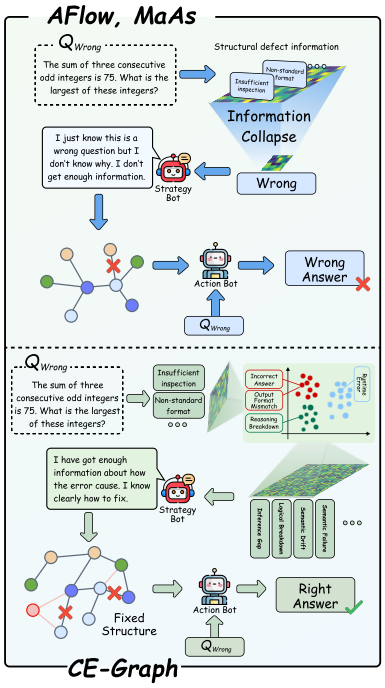


Figure 1: Upper: traditional LLMs compress rich error traces into binary signals, causing "information collapse" and obscuring the underlying failure distribution, which hinders systematic refinement. Lower: **CE-Graph** leverages the Failure Signature Space to cluster errors into coherent modes

2.2 THE PITFALLS OF GLOBAL SEARCH

The dominant global search paradigm, exemplified by methods like Monte Carlo Tree Search (MCTS) in systems such as MaAS (Zhang et al., 2025a) or AFlow (Zhang et al., 2025b), attempts to minimize failure mass indirectly. It relies on a global performance metric $G(W, D)$, which approximates $1 - M(W)$ via black-box (Huang et al., 2024; Cheng et al., 2024) sampling, i.e., essentially averaging success rates over a dataset D. The objective is thus to maximize this score: $W^* = \arg\max_{W \in S} G(W, D)$.

Although intuitive, this approach suffers from a critical flaw: information collapse. The optimizer accesses only a collapsed binary signal for each execution trace τ_d, i.e., $\text{Observe}(\tau_d) \in \{0, 1\}$, discarding the rich details of why and where the failure occurred. For example, in code generation tasks like HumanEval (Chen et al., 2021), multiple failures might stem from similar semantic issues (e.g., off-by-one errors), but global search treats them as independent binary outcomes, preventing the modeling of underlying patterns. It lacks details about the failure's location $\mathbf{s}$ in $\mathcal{F}$, preventing estimation of $p(\mathbf{s} \mid W)$'s shape. Essentially, the optimizer is blindfolded, unable to detect "mountains" on the failure map and forced to wander randomly. This reduces $M(W)$ to an opaque scalar, resulting in inefficient zero-order optimization in a vast search space.

Quantitatively, this inefficiency manifests as high sample complexity: global methods often require thousands of evaluations to converge, as they cannot exploit failure structure (Snoek et al., 2012). In contrast, our paradigm leverages distributional insights to achieve faster, more targeted improvements, as detailed in Section 2.3.

2.3 THE FAILURE-DRIVEN REFINEMENT PARADIGM

To efficiently minimize the total failure mass, the optimizer must estimate the landscape of $p(\mathbf{s} \mid W)$ and take steps that maximally reduce its mass. This forms the core of our paradigm, which transforms optimization from a global, brute-force search into a local, iterative refinement process. Instead of a single global search, we reframe optimization as an iterative process, progressively refining the workflow via the update rule $W_{t+1} = W_t \oplus \Delta_t$. This draws inspiration from counterexample-guided methods in formal verification but adapts them to the stochastic, semantic nature of LLM failures (Debbi, 2018; Shorinwa et al., 2025).

The key is selecting Δ_t. Since global minimization is intractable in one step, due to the combinatorial explosion of possible workflows, we adopt a greedy (Nemhauser et al., 1978; Vazirani, 2001) approach: at each step t, we seek the local edit that maximally reduces failure mass. This yields the theoretical objective:

$$\Delta_t = \underset{\Delta \in \mathcal{A}(W_t, \mathcal{O})}{\arg\max} \left[M(W_t) - M(W_t \oplus \Delta) \right] = \underset{\Delta \in \mathcal{A}(W_t, \mathcal{O})}{\arg\max} \left[\int_{\mathcal{F}} p(\mathbf{s} \mid W_t)\, d\mathbf{s} - \int_{\mathcal{F}} p(\mathbf{s} \mid W_t \oplus \Delta)\, d\mathbf{s} \right]. \tag{2}$$

Directly solving this functional optimization is intractable, but it provides a clear blueprint for approximation. By focusing on mass reduction, this objective enables a "gradient-like" descent, where each edit targets dense failure regions, leading to more efficient convergence compared to random sampling in global search.

2.3.1 THEORETICAL PROPERTIES OF GREEDY MASS REDUCTION

Under mild assumptions, our greedy approach guarantees progressive reduction in failure mass, bridging formal methods like CEGAR (Hidvégi et al., 2024) with stochastic LLM optimization. Assume that $p(\mathbf{s} \mid W)$ is Lipschitz continuous with constant L and that edits Δ have bounded impact (i.e., $\|p(\mathbf{s} \mid W \oplus \Delta) - p(\mathbf{s} \mid W)\|_\infty \leq B$).

Theorem 1 (Greedy Reduction Bound)**.** *Let Δ_t be selected as in Eq. 2. If the edit reduces the mass in the target mode by at least $\delta > 0$, then $M(W_{t+1}) \leq M(W_t) - \delta + \epsilon$, where $\epsilon = O(L \cdot B \cdot \mu(\mathcal{F} \setminus b_t^*))$ bounds spillover effects to non-target regions (with μ denoting the measure of the space).*

Proof Sketch. Decompose the integral in Eq. 2 over modes: $M(W) = \int_{b_t^*} p(\mathbf{s} \mid W)\, d\mathbf{s} + \int_{\mathcal{F} \setminus b_t^*} p(\mathbf{s} \mid W)\, d\mathbf{s}$. The greedy edit ensures the first term decreases by at least δ. Spillover to other modes is bounded by Lipschitz continuity and the edit's impact, yielding ϵ. The full proof, including convergence to a stationary point under repeated application, is provided in Appendix G. □

3 CE-GRAPH: A PRACTICAL ALGORITHM FOR FAILURE MASS REDUCTION

We detail our CE-Graph to approximate the intractable greedy failure mass reduction objective (Eq. 2) (Clarke et al., 2000), transforming abstract concepts into actionable steps. By deconstructing the challenge into three principled stages, i.e., empirical density estimation, gradient approximation, and vetted state transition, we operationalize a greedy descent on the failure landscape (Bishop, 2006). This not only approximates the theoretical ideal but also addresses real-world challenges like stochastic failures and large edit spaces (Zhang et al., 2025b).

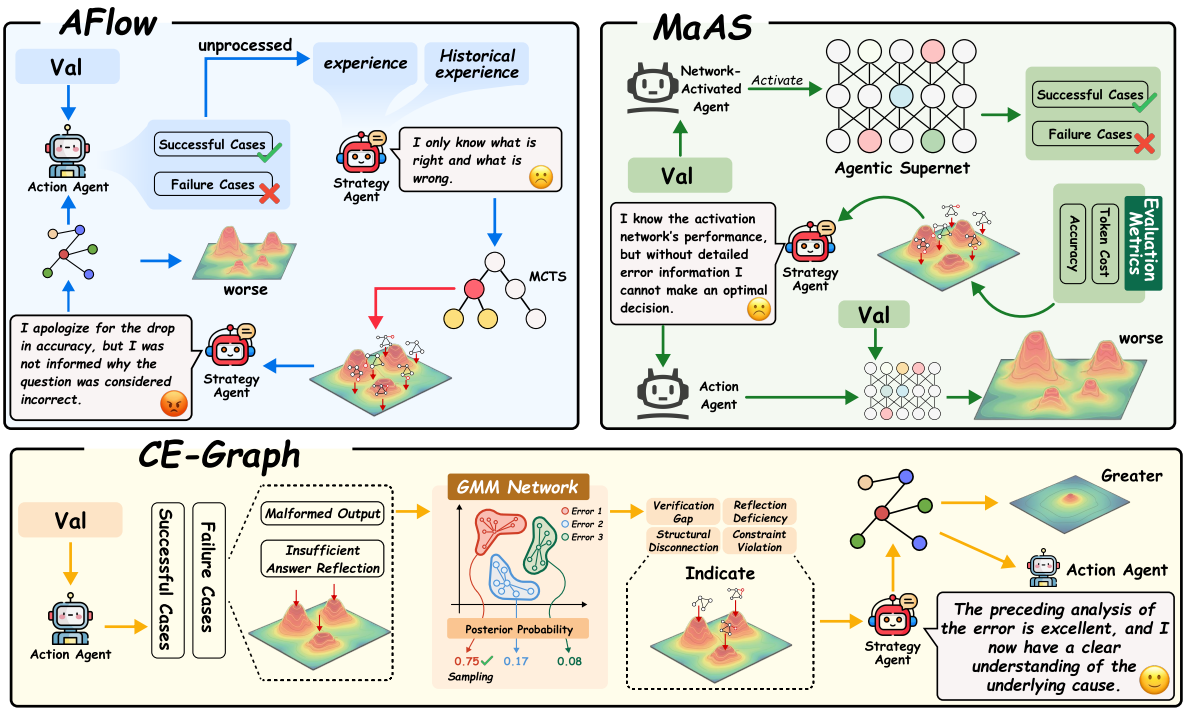


Figure 2: Overview of our CE-Graph framework. The process iteratively refines workflows by (i) distilling raw failure traces into structured signatures, (ii) clustering to expose dense failure modes, and (iii) applying targeted Propose-and-Verify graph edits, enabling principled descent on the failure landscape.

The following subsections elaborate on each stage's role, including a detailed analysis of their approximations and alignments with the theory.

3.1 Constructing the Failure Signature Space $\mathcal{F}$

The first step transforms unstructured data in the Counterexample Pool C_t into a tractable format, enabling empirical estimation of $p(\mathbf{s} \mid W_t)$. Without this, failures remain opaque, as in global search methods. We define a mapping ϕ that projects a raw trace τ_d into the structured **Failure Signature Space** $\mathcal{F}$ through two stages, ensuring both structural and semantic fidelity.

First, **Failure Distillation** uses a utility LLM to distill τ_d into a concise tuple $(v_{\text{err}}, z_{\text{err}})$, capturing the error-producing node and message. This step compresses verbose traces (e.g., a multi-step math failure involving incorrect LCM calculation) into analyzable components. Second, **Semantic-Structural Vectorization** maps this tuple to the signature vector $\mathbf{s}$, preserving orthogonality and similarity.

Formally, let V be the set of workflow graph nodes and $\mathcal{Z}$ the space of error messages. We define: - A **structural mapping** $\psi_{\text{struct}} : V \to \mathbb{R}^{|V|}$, yielding a one-hot vector for a node identifier. - A **semantic mapping** $\psi_{\text{sem}} : \mathcal{Z} \to \mathbb{R}^d$, embedding an error message into d dimensions (e.g., using BERT-like models for semantic clustering).

The final failure signature is:

Definition 1 (Failure Signature Vector)**.** *A failure signature* $\mathbf{s} \in \mathcal{F}$ *for a distilled trace* (v_{err}, z_{err}) *is:*

$$\mathbf{s} = \phi(\tau_d) = \psi_{struct}(v_{err}) \oplus \psi_{sem}(z_{err}), \tag{3}$$

where $\oplus$ *denotes vector concatenation.*

The structural component $\psi_{\text{struct}}(v_{\text{err}})$ identifies *where* the failure occurred, mapping to orthogonal subspaces in $\mathcal{F}$ for node-specific patterns, crucial for attributing blame in multi-agent workflows. The semantic component $\psi_{\text{sem}}(z_{\text{err}})$ captures *what* error occurred, grouping semantically similar messages (e.g., "calculation error" vs. "the sum was incorrect"). This hybrid embedding ensures that the resulting point cloud $S_t = \{\phi(\tau_d) \mid \tau_d \in C_t\}$ possesses an informative geometry, providing a robust empirical basis for approximating $p(\mathbf{s} \mid W_t)$. In practice, this construction mitigates information loss, allowing CE-Graph to detect clusters that global methods overlook.

3.2 Solving the Refinement Objective

CE-Graph approximates the greedy mass reduction by two steps: approximating the gradient direction, then finding an optimal edit. This decomposition balances computational tractability with theoretical fidelity, ensuring each iteration reduces $M(W_t)$ as per Theorem 1.

3.2.1 Step 1: Approximating the Gradient Direction

We approximate the failure mass gradient by identifying the densest region in S_t's empirical distribution, which corresponds to the most prevalent failure mode. This step operationalizes the "steepest descent" in our paradigm, focusing repairs on high-mass areas. Using non-parametric density estimation, we fit a Gaussian Mixture Model (GMM) with parameters θ to S_t. Each component b_k represents a distinct failure mode (e.g., a cluster of division-by-zero errors in code tasks), with prevalence π_k. The most prevalent mode b_t^* maximizes the mixing coefficient:

$$b_t^* = \arg\max_{b_k \in B_t} \pi_k = \arg\max_{b_k \in B_t} \sum_{\mathbf{s}_i \in S_t} p(b_k \mid \mathbf{s}_i, \theta), \tag{4}$$

where $p(b_k \mid \mathbf{s}_i, \theta)$ is the posterior probability. This b_t^* approximates the steepest descent direction, guiding repairs toward systemic flaws rather than isolated errors.

3.2.2 Analysis of Gradient Approximation

The GMM provides a consistent estimator of $p(\mathbf{s} \mid W_t)$ under standard conditions (e.g., as $|S_t| \to \infty$, using BIC for model selection), making it a reliable proxy for the true density. The densest mode b_t^* approximates the global mode of maximum density, with estimation error bounded by $O(1/\sqrt{|S_t|})$ in expectation (cf. kernel density estimation theory). This ensures that our approximation aligns with the theoretical greedy objective in Eq. 2, with provable convergence as counterexamples accumulate. In benchmarks like MATH (Hendrycks et al., 2021), where failures cluster around algebraic misconceptions, this approximation effectively identifies "mountains" that reduce overall mass, outperforming uniform sampling by focusing on high-impact modes.

3.2.3 Step 2: Finding the Optimal Edit (Δ_t) via Propose-and-Verify

This step addresses two challenges: the vast edit space $\mathcal{A}(W_t, \mathcal{O})$ (potentially exponential in workflow size) and the unknown utility distribution over instances. Our **Propose-and-Verify** mechanism handles them in two stages, combining LLM creativity with empirical validation to approximate the argmax in Eq. 2.

Constrained Proposal: A "Proposer" LLM, conditioned on b_t^*'s summary (e.g., "recurring off-by-one errors in loops"), generates N diverse high-potential edits $\{\Delta_1, \ldots, \Delta_N\}$, reducing the search to a promising subset. This heuristic leverages LLM's generative power while constraining outputs to the operator library $\mathcal{O}$, ensuring feasibility. **Verification:** We estimate each candidate's utility by sampling K instances from b_t^* and computing the empirical success rate:

$$V(\Delta_i) \approx \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\left[\text{Verify}(\text{Execute}(W_t \oplus \Delta_i, x_k), y_k) = 1\right]. \tag{5}$$

The highest $V(\Delta_i)$ yields Δ_t, enabling efficient, evidence-based solutions. This Monte Carlo approach provides an unbiased estimator of mass reduction, with variance decreasing as $O(1/K)$, aligning with our bound in Theorem 1. Further details are provided in Appendix H.

4 Experiments

We benchmark our failure-driven refinement paradigm against a comprehensive suite of baselines across mathematical reasoning, code generation, and complex tool-use tasks to validate its superior robustness and efficiency.**Experiment Setup Tasks and Benchmarks.** We evaluate CE-Graph on six public benchmarks across three domains: (1) **math reasoning** (GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), MultiArith (Roy & Roth, 2015)); (2) **code generation** (HumanEval, MBPP) (Chen et al., 2021; Austin et al., 2021); and (3) **tool use** (GAIA) (Mialon et al., 2023). This diverse task set allows for a thorough assessment of our workflow's versatility. **Baselines.** We compare CE-Graph against three categories of agentic systems: (1) **single agent execution** (Vanilla, CoT (Wei et al., 2022), ComplexCoT (Kojima et al., 2022), SC (Wang et al., 2023)); (2) **hand-crafted multi-agent systems** (MultiPersona (Wang et al., 2024), LLM-Debate (Du et al., 2024), LLM-Blender (Jiang et al., 2023), DyLAN (Liu et al., 2024b), AgentVerse (Chen et al., 2024b), MacNet (Qian et al., 2025)); and (3) **automated agentic systems** (AutoAgents (Chen et al., 2024a), GPTSwarm, ADAS (Hu et al., 2025), AgentSquare (Shang et al., 2025), AFlow (Zhang et al., 2025b), and the previous state-of-the-art, MaAS (Zhang et al., 2025a)). **Implementation Details.** All experiments use `gpt-40-mini (OpenAI et al., 2024)` as the base LLM. CE-Graph starts from a simple workflow, stops at validation convergence or 20 iterations, and uses an operator library $\mathcal{O}$ with `RevisePrompt`, `InsertNode`, and `DeleteNode`.

Algorithm 1 The CE-Graph Algorithm

Input: Initial workflow W_0, Dataset D, Hyperparameters $N, K, k, \epsilon, T_{\max}$
Output: Optimized workflow W_T
Initialize $t \leftarrow 0$, validation scores $\mathcal{G} \leftarrow \emptyset$
while not converged **and** $t < T_{\max}$ **do**
    // Sample failure distribution $p(\mathbf{s} \mid W_t)$
    Populate Counterexample Pool C_t by executing W_t on D.
    if C_t is empty **then** ▷ No failures; optimization complete
        break
    end if
    Construct embedding $S_t = \{\phi(\tau_d) \mid (d, \tau_d) \in C_t\}$.
// Approximate greedy mass reduction (Eq. 2)
    // Step 1: Gradient direction
    Fit GMM to S_t for modes B_t and θ.
    Identify densest mode b_t^* using Eq. 4.
    // Step 2: Optimal edit
    Propose candidates $\{\Delta_1, \ldots, \Delta_N\} \subset \mathcal{A}(W_t, \mathcal{O})$ for b_t^*.
    Select $\Delta_t \leftarrow \arg\max_{\Delta_i} V(\Delta_i)$ via Monte Carlo.
    Apply: $W_{t+1} \leftarrow W_t \oplus \Delta_t$.
// Evaluation and check
    Evaluate g_t on validation set; update $\mathcal{G}$.
    $t \leftarrow t + 1$
end while
return W_t

4.1 Performance Analyses

As shown in Table 1, CE-Graph consistently outperforms all baselines across every task domain and establishes a new state-of-the-art average score of **86.23%**, widening the gap over the previous best of 83.59% achieved by MaAS. The gains are substantial across categories: on the **MATH** benchmark, CE-Graph improves accuracy by +4.1%; on the **MBPP** code synthesis task, it surpasses MaAS by +5.9%; and on **HumanEval**, it delivers an additional +1.4% improvement. Together, these results demonstrate that CE-Graph not only strengthens mathematical reasoning but also enhances program synthesis and tool-use capabilities in a balanced manner. This broad-based improvement strongly validates the effectiveness of our counterexample-guided optimization. Our failure-driven refinement paradigm proves especially advantageous in complex, multi-domain environments such as the GAIA benchmark (Figure 3). In these scenarios, monolithic systems like AFlow and ADAS struggle to generalize because a single static workflow cannot adequately cover heterogeneous task requirements. In contrast, CE-Graph iteratively detects and repairs the most prevalent failure modes, evolving into a single yet highly robust workflow capable of adapting across domains. This explains the substantial improvements observed at all difficulty levels, with the largest gains on Level 3 tasks, underscoring CE-Graph's ability to build truly general-purpose agentic systems. Beyond absolute scores, CE-Graph also demonstrates notable consistency. Whereas many baselines fluctuate across tasks, our approach delivers high and stable performance in math reasoning, program synthesis, and tool use alike. Moreover, its advantage grows with task difficulty: while competing methods plateau or even regress when facing long-horizon or error-prone problems, CE-Graph maintains steady gains by systematically targeting recurring failure modes. Taken together, these findings establish CE-Graph as not only the most accurate but also the most reliable and scalable framework for optimizing agentic workflows.

4.2 Cost Analyses

Beyond raw performance, CE-Graph is exceptionally resource-efficient. Our cost analysis on the **MATH** benchmark (Figure 3, right) reveals that CE-Graph achieves the highest accuracy (53.5%) while maintaining remarkably low computational costs. This result validates the central theoretical hypothesis behind CE-Graph: by shifting from inefficient global search to targeted, counterexample-guided local structure editing, our CE-Graph eliminates redundant sampling and costly trial-and-error explorations. This principled approach directly reduces training token consumption and monetary costs without compromising optimization convergence. In addition, CE-Graph demonstrates strong robustness to token budget sensitivity: accuracy remains stable even under restricted optimization budgets, whereas baselines such as AFlow and MaAS exhibit steep declines due to their dependence on large-scale search. The convergence-aware stopping criterion is particularly effective, yielding over 50% savings in optimization cost by halting iterations once the failure distribution has stabilized. Ablation results further show that structured operators and verification both play key roles in ensuring efficient resource usage, since removing them leads to higher token consumption with markedly lower accuracy:contentReference[oaicite:0]index=0.

Table 1: A complete performance comparison on math and code benchmarks. Results are highlighted using shades of gray for the header and top performers, bold for the best, and underlining for the runner-up.

Method	GSM8K	MATH	MultiArith	HumanEval	MBPP	Avg.
Single Agent Execution						
Vanilla (Baseline)	87.45	46.29	96.85	87.08	71.83	77.50
CoT (Wei et al., 2022)	87.10	46.40	96.31	88.13	71.83	77.95
ComplexCoT (Kojima et al., 2022)	86.89	46.53	96.70	87.49	72.36	78.00
SC (Wang et al., 2023) (CoT×5)	87.57	47.91	96.58	88.60	73.60	78.85
Hand-craft Multi-agent Systems						
MultiPersona (Wang et al., 2024)	87.50	45.43	97.49	88.32	73.19	78.39
LLM-Debate (Du et al., 2024)	89.47	48.54	97.33	88.68	70.29	78.86
LLM-Blender (Jiang et al., 2023)	88.35	46.92	97.29	88.80	77.05	79.68
DyLAN (Liu et al., 2024b)	89.98	48.63	97.12	90.42	77.30	80.69
AgentVerse (Chen et al., 2024b)	89.91	47.35	97.50	89.29	74.28	79.67
MacNet (Qian et al., 2025)	87.95	45.18	96.03	84.57	65.28	75.00
Automated Agentic Systems						
AutoAgents (Chen et al., 2024a)	87.69	45.32	96.42	87.64	71.95	77.80
GPTSwarm (Zhuge et al., 2024)	89.14	47.88	96.79	89.32	77.43	80.11
ADAS(Hu et al., 2025)	86.12	43.18	96.02	84.19	68.13	75.13
AgentSquare(Shang et al., 2025)	87.62	48.51	97.77	89.08	78.46	80.29
AFlow(Zhang et al., 2025b)	91.16	51.28	96.22	90.93	81.67	82.25
MaAS(Zhang et al., 2025a)	92.30	51.82	98.80	92.85	82.17	83.59
CE-Graph (Ours)	**93.70**	**55.91**	**99.20**	**94.26**	**88.10**	**86.23**

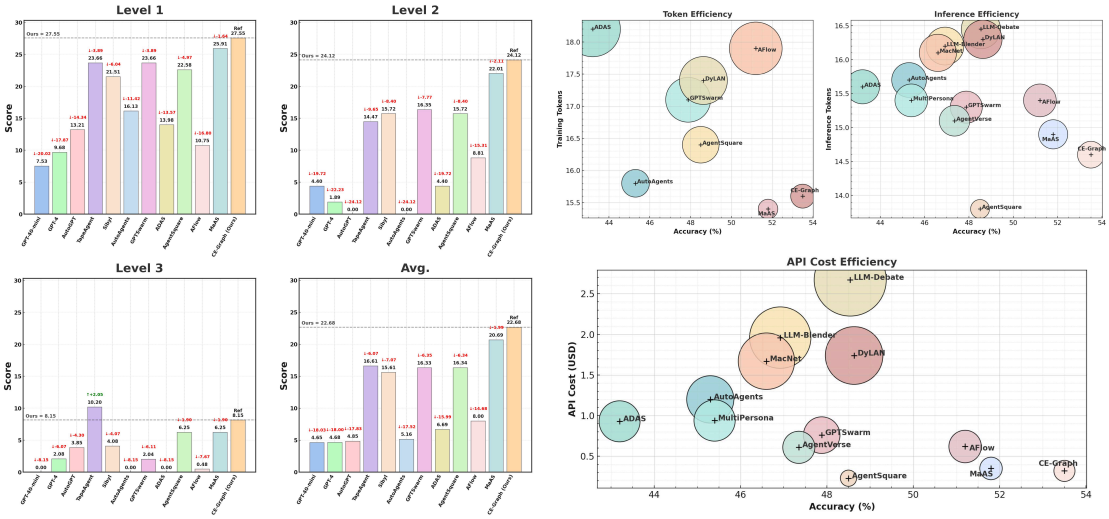


Figure 3: **CE-Graph** outperforms all baselines on GAIA (Levels 1–3, Avg.) and achieves the best accuracy–efficiency trade-off (tokens, API cost) in the ideal lower-right region.

4.3 Clustering-Driven Correction Stability

To precisely evaluate the stability and sustained refinement capability of our CE-Graph, we design a longitudinal assessment aimed at verifying its long-term effectiveness in locating and resolving fundamental failures. We begin by executing an unoptimized baseline workflow (W_0) on three mathematical reasoning benchmarks, i.e., GSM8K, MATH, and MultiArith, and collect all resulting failure cases to construct a fixed initial failure set (E_0). During the subsequent 20 rounds of optimization, each updated workflow (W_t) is re-evaluated on this fixed set (E_0). We continuously track a single core metric: *Accuracy on* E_0, defined as the proportion of initial failure examples successfully repaired by W_t. This fixed-set evaluation design effectively isolates variables, thereby providing a clear lens through which to assess the refinement mechanism's inherent capacity for stable and accumulative improvements over time. The longitudinal evaluation conducted on fixed failure sets (E_0) across the three mathematical reasoning benchmarks clearly demonstrates the superior stability and sustained improvement capability of our CE-Graph. As shown in Figure 4, CE-Graph yields a smooth and monotonically increasing performance trajectory on all benchmarks, indicating a robust and accumulative optimization process. In stark contrast, baseline methods, particularly AFlow, exhibit pronounced fluctuations, revealing a tendency toward *policy oscillation*, where localized repairs inadvertently disrupt previously correct solutions. This instability reflects the classical is-

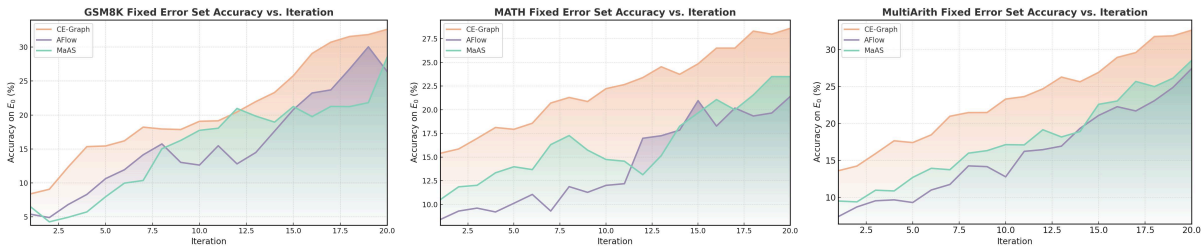


Figure 4: The refinement accuracy on the fixed failure set (E_0) over 20 optimization rounds across three mathematical reasoning benchmarks: GSM8K, MATH, and MultiArith. **CE-Graph** (orange) demonstrates a smooth and monotonically increasing trajectory on all datasets, highlighting the stability and accumulative effect of its refinement process. In contrast, baseline methods, especially **AFlow** (purple), exhibit significant performance fluctuations, reflecting instability induced by ad-hoc refinement strategies and policy oscillation.

sue of *information collapse*, wherein structural correlations among failures are not preserved or exploited, leaving the refinement process unguided. CE-Graph effectively overcomes this challenge via failure clustering and a principled propose-and-verify mechanism, transforming discrete failure signals into coherent insights about systemic defects. This ensures that each iteration constitutes a net-positive improvement.

4.4 Ablation Studies

To rigorously assess the contribution of each core component within our CE-Graph, we conduct a comprehensive ablation study. Specifically, we treat the full CE-Graph model as the performance reference and systematically remove four key mechanisms to construct comparison variants: (1) **w/o Clustering**, which removes the failure clustering module to evaluate the importance of identifying systematic failure patterns; (2) **w/o Verification**, which disables the verification step in the propose-and-verify mechanism to quantify its role in ensuring the quality of refinement; (3) **w/o Structured Operators**, which removes the structured operator library to highlight the critical role of a structured action space in generating effective repairs; and (4) **w/o Convergence-aware Stopping**, which omits the adaptive convergence-aware stopping criterion to measure its efficiency in reducing optimization cost. All models are evaluated on both MATH and HumanEval benchmarks. We report Accuracy and pass@1 as performance metrics, and track the optimization token consumption (Cost) to ensure a fair and comprehensive comparison. Beyond raw scores, we also examine the optimization dynamics to reveal how different ablations alter the repair trajectories. This dual perspective not only quantifies the magnitude of performance degradation, but also clarifies the distinct role each module plays—whether in boosting accuracy, improving efficiency, or stabilizing convergence. Taken together, these analyses confirm that CE-Graph's overall gains stem from the synergistic integration of its modules rather than from any isolated component.

Table 2: Ablation study of CE-Graph on MATH and HumanEval. Accuracy (%) and pass@1 (%) measure performance; Cost denotes optimization tokens (10^3).

	MATH		HumanEval	
Method	Accuracy (%)	Cost (10^3)	pass@1 (%)	Cost (10^3)
CE-Graph (Full Model)	**55.91**	**1,210**	**94.26**	**955**
w/o Clustering	51.25	1,190	91.50	940
w/o Verification	49.10	1,250	89.75	980
w/o Structured Operators	**47.35**	**1,310**	**87.20**	**1,050**
w/o Convergence-aware Stopping	53.50	1,850	93.80	1,520

As shown in Table 2, our ablation results clearly reveal the synergy and indispensability of the components within our CE-Graph. Among all variants, the most significant performance degradation is observed in **w/o Structured Operators**, with accuracy dropping to 47.35% on MATH and 87.20% on HumanEval. This result strongly demonstrates that merely identifying failure patterns is insufficient; a discrete and effective action space defined by a structured operator library is essential for generating reliable repairs. The second most severe degradation is observed in the **w/o Verification** variant, highlighting the verification stage as a critical safeguard. It effectively filters hallucinations from the Proposer model and ensures that each optimization step constitutes a validated improvement. In addition, removing failure clustering (**w/o Clustering**) also leads to a notable performance drop, validating our core premise: targeting widespread, structural failure modes is significantly more effective than addressing isolated, stochastic errors. Finally, removing the convergence-aware stopping criterion (**w/o Convergence-aware Stopping**) leads to a slight performance drop while increasing the optimization cost by more than 50%, underscoring the module's substantial value in improving efficiency. Beyond these individual effects, the ablation study also provides a holistic view of how different modules interact: for example, clustering and structured operators jointly enable mode-specific proposals, while verification and convergence-aware stopping collectively ensure stability and efficiency. This interplay illustrates that the design of CE-Graph

follows a tightly coupled workflow rather than an additive composition. In summary, these findings collectively confirm that CE-Graph's superior performance arises not from any single component, but from the tight integration and synergy among its modules, i.e., structured diagnosis, principled proposal, and empirical verification, ultimately establishing a robust and generalizable paradigm for counterexample-guided optimization.

5 Related Works

Automated Optimization of Agentic Workflows Automating multi-agent workflow optimization is a growing field, with methods like MaAS (Zhang et al., 2025a), AFlow (Zhang et al., 2025b), AgentSquare (Shang et al., 2025), and AdaptFlow (Zhu et al., 2025) using global search strategies such as Monte Carlo Tree Search (MCTS) or evolutionary algorithms to maximize success rates. Frameworks like DSPy (Khattab et al., 2024) optimize prompts with scalar metrics but suffer from "information collapse," reducing rich failure traces to binary signals and limiting optimizers to inefficient zero-order methods. CE-Graph stands out by preserving both the semantics and structure of failures via its Failure Signature Space, shifting to a superior Expected Failure Mass minimization approach. This white-box, distributional perspective enables cost-effective repairs and delivers substantially higher robustness, marking a step toward more interpretable and scalable workflow optimization in agentic systems.

Counterexample-Guided Refinement in Formal Methods Counterexample-guided techniques, rooted in program synthesis and formal verification (e.g., CEGAR (Hidvégi et al., 2024)), iteratively refine models using specification-violating examples, with adaptations like CigaR (Renze & Guven, 2024) extending to stochastic settings. However, these methods target deterministic systems and struggle to cope with the stochastic and semantic nature of LLM workflow failures (Chang et al., 2024). CE-Graph innovates by adapting this principle to open-ended LLM domains, replacing formal verification with empirical failure-mode clustering and constrained graph edits. This shift allows CE-Graph to resolve subtle reasoning flaws that are invisible to symbolic checkers, yielding more generalizable and efficient reliability improvements compared to classical counterexample-guided refinement. In doing so, CE-Graph effectively bridges the rigor of formal refinement with the flexibility required for large-scale, data-driven agentic systems.

Instance-Level Self-Correction and Reflection Self-correction methods like Self-Consistency (Wang et al., 2023), Reflexion (Shinn et al., 2023), and Self-Refine (Madaan et al., 2023) use voting or prompt-based reflection to improve individual outputs, with studies like (Renze & Guven, 2024) and benchmarks like AgentBench (Liu et al., 2024a) exploring agent reasoning. While these techniques can be effective in isolated cases, they often risk overfitting and fail to capture systemic flaws. CE-Graph instead adopts a global distributional view, aggregating counterexamples into failure clusters and applying structured repairs that address recurring error modes. This global optimization perspective ensures stability across iterations and scalability to diverse domains, making CE-Graph a more principled and reliable solution for complex agentic systems. By shifting the focus from local instance-level fixes to systemic distributional refinement, CE-Graph offers a complementary direction that substantially broadens the scope of reliability research in LLM-based workflows.

6 Discussions and Limitations

While CE-Graph demonstrates strong improvements in both robustness and efficiency, several limitations remain. Firstly, the construction of the Failure Signature Space relies on semantic embeddings that may not perfectly capture subtle reasoning errors; misaligned embeddings can blur distinct failure modes. Second, our refinement strategy assumes that the most prevalent failure clusters correspond to the most critical modes to fix. Although effective in practice, this greedy prioritization may overlook rare but catastrophic errors. Third, our ablation study highlights that the operator library $\mathcal{O}$ plays a decisive but delicate role in refinement effectiveness. On the one hand, reducing $\mathcal{O}$ to a minimal set (w/o Structured Operators) leads to drastic performance degradation, confirming that a structured action space is indispensable for meaningful edits. On the other hand, enlarging $\mathcal{O}$ without control risks reintroducing the inefficiency of global search, since a richer operator pool exponentially increases the proposal space and the chance of spurious or low-utility edits. This reveals a fundamental trade-off: operators must be expressive enough to capture diverse failure modes, yet sufficiently constrained to avoid combinatorial explosion. Designing operator libraries that adaptively expand with observed failure diversity, or that leverage hierarchical abstractions, may offer a promising route forward. Finally, although our convergence-aware stopping rule mitigates over-optimization, it does not guarantee global optimality, and premature stopping may occasionally freeze workflows in suboptimal states. Despite these limitations, we believe CE-Graph establishes a principled direction for workflow optimization. Future extensions could incorporate (i) adaptive embeddings that evolve with the workflow, (ii) risk-sensitive objectives that balance frequent and rare failures, and (iii) meta-learning strategies that dynamically expand or prune the operator library. Exploring these directions may further improve both the theoretical guarantees and the practical scalability of failure-driven refinement.

7 ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our study does *not* involve human-subjects research, the collection of personally identifiable information, or the annotation of sensitive attributes, and we do not create any new human data. All experiments are conducted strictly on publicly available and widely used vision–language and reasoning benchmarks, in compliance with their respective licenses and terms of use. We ensure that our methodology and reported results are focused solely on advancing technical understanding and do not raise ethical risks associated with data privacy or human annotation practices.

8 REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of our results. The paper provides detailed descriptions of our model architecture, algorithmic components, and hyperparameter settings. All benchmarks used in our experiments are publicly available, and we clearly specify their versions and evaluation protocols. Furthermore, we outline the ablation settings, stopping criteria, and cost analysis to enable faithful reproduction of our findings. Pseudocode for our main algorithm is included, and we will release implementation details and experiment scripts to facilitate replication and further research upon acceptance.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 978-0-387-31073-2. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3), March 2024. ISSN 2157-6904. doi: 10.1145/3641289. URL https://doi.org/10.1145/3641289.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. Autoagents: A framework for automatic agent generation, 2024a. URL https://arxiv.org/abs/2309.17288.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=EHg5GDnyq1.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3201–3219, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.176. URL https://aclanthology.org/2024.acl-long.176/.

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. pp. 154–169, 2000. doi: 10.1007/10722167_15. URL https://link.springer.com/chapter/10.1007/10722167_15.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of the 5th International Conference on Computers and Games*, pp. 72–83, 2006.

Hichem Debbi. Counterexamples in model checking: A survey. *Informatica*, 42(2):145–166, 2018. URL https://www.informatica.si/index.php/informatica/article/view/1442.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Dávid Hidvégi, Khashayar Etemadi, Sofia Bobadilla, and Martin Monperrus. Cigar: Cost-efficient program repair with llms, 2024. URL https://arxiv.org/abs/2402.06598.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Beichen Huang, Xingyu Wu, Yu Zhou, Jibin Wu, Liang Feng, Ran Cheng, and Kay Chen Tan. Exploring the true potential: Evaluating the black-box optimization capability of large language models. *CoRR*, abs/2404.06290, 2024. URL https://doi.org/10.48550/arXiv.2404.06290.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2), January 2025. ISSN 1046-8188. doi: 10.1145/3703155. URL https://doi.org/10.1145/3703155.

Shankar Kumar Jeyakumar, Alaa Alameer Ahmad, and Adrian Garret Gabriel. Advancing agentic systems: Dynamic task decomposition, tool integration and evaluation using novel metrics and dataset. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024. URL https://openreview.net/forum?id=kRRLhPp7CO.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines. 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. Autoflow: Automated workflow generation for large language model agents, 2024. URL https://arxiv.org/abs/2407.12821.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic LLM-agent network: An LLM-agent collaboration framework with agent team optimization, 2024b. URL https://openreview.net/forum?id=i43XCU54Br.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2025. URL https://arxiv.org/abs/2402.06196.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978. doi: 10.1007/BF01588971.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, and Kevin Button. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Melissa Z Pan, Mert Cemri, Lakshya A Agrawal, Shuyi Yang, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Kannan Ramchandran, Dan Klein, Joseph E. Gonzalez, Matei Zaharia, and Ion Stoica. Why do multiagent systems fail? In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=wM521FqPvI.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=K3n5jPkrU6.

Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pp. 476–483. IEEE, November 2024. doi: 10.1109/fllm63129.2024.10852493. URL http://dx.doi.org/10.1109/FLLM63129.2024.10852493.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442/.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In Lluís Màrquez, Chris Callison-Burch, and Jian Su (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *Information Fusion*, 126:103599, February 2026. ISSN 1566-2535. doi: 10.1016/j.inffus.2025.103599. URL http://dx.doi.org/10.1016/j.inffus.2025.103599.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space, 2025. URL https://arxiv.org/abs/2410.06153.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *ACM Comput. Surv.*, 58(3), September 2025. ISSN 0360-0300. doi: 10.1145/3744238. URL https://doi.org/10.1145/3744238.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012. URL https://arxiv.org/abs/1206.2944.

Peiyang Song, Pengrui Han, and Noah Goodman. A survey on large language model reasoning failures. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL https://openreview.net/forum?id=hsgMn4KBFG.

Benjamin Steenhoek, Md Mahbubur Rahman, Monoshi Kumar Roy, Mirza Sanjida Alam, Hengbo Tong, Swarna Das, Earl T. Barr, and Wei Le. To err is machine: Vulnerability detection challenges llm reasoning, 2025. URL https://arxiv.org/abs/2403.17218.

Maciej undefinedwiechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: a review of recent modifications and applications. *Artif. Intell. Rev.*, 56(3):2497–2562, July 2022. ISSN 0269-2821. doi: 10.1007/s10462-022-10228-y. URL https://doi.org/10.1007/s10462-022-10228-y.

Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Vijay V. Vazirani. *Approximation Algorithms*. Springer, Berlin, Heidelberg, 2001. ISBN 978-3-540-65367-7. doi: 10.1007/978-3-662-04565-7. URL https://en.wikipedia.org/wiki/Approximation_algorithm.

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

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 257–279, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.15. URL https://aclanthology.org/2024.naacl-long.15/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Chaojia Yu, Zihan Cheng, Hanwen Cui, Yishuo Gao, Zexu Luo, Yijin Wang, Hangbin Zheng, and Yong Zhao. A survey on agent workflow – status and future. In *2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 770–781, 2025a. doi: 10.1109/ICAIBD64986.2025.11082076.

Chaojia Yu, Zihan Cheng, Hanwen Cui, Yishuo Gao, Zexu Luo, Yijin Wang, Hangbin Zheng, and Yong Zhao. A survey on agent workflow – status and future. In *2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 770–781. IEEE, May 2025b. doi: 10.1109/icaibd64986.2025.11082076. URL http://dx.doi.org/10.1109/ICAIBD64986.2025.11082076.

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

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=z5uVAKwmjf.

Jusheng Zhang, Zimeng Huang, Yijia Fan, Ningyuan Liu, Mingyan Li, Zhuojie Yang, Jiawei Yao, Jian Wang, and Keze Wang. KABB: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems. In *Forty-second International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=AKvy9a4jho.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Forty-second International Conference on Machine Learning*, 2025d. URL https://openreview.net/forum?id=GazlTYxZss.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025. URL https://arxiv.org/abs/2303.18223.

Runchuan Zhu, Bowen Jiang, Lingrui Mei, Fangkai Yang, Lu Wang, Haoxiang Gao, Fengshuo Bai, Pu Zhao, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Adaptflow: Adaptive workflow optimization via meta-learning, 2025. URL https://arxiv.org/abs/2508.08053.

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

A IMPLEMENTATION DETAILS FOR FAILURE DIAGNOSIS

The entire CE-Graph framework relies on the quality of its initial diagnosis. Therefore, the reliability of the **Failure Distillation** step (introduced in Section 3.1) is paramount. We engineer its robustness through two primary mechanisms that leverage the strengths of LLMs while mitigating their weaknesses.

1. Constrained Task Formulation. We deliberately avoid tasking the utility LLM with open-ended generation, a high-entropy process known to be susceptible to hallucination. Instead, we frame distillation as a highly constrained, low-entropy information extraction task. The LLM is prompted to function as a pattern recognizer, identifying a specific node from a known graph and extracting an error message from a given text trace. This leverages the well-established strengths of modern LLMs in structured data processing and classification, ensuring high-fidelity outputs.

2. Statistical Fault Tolerance. Our framework is designed with statistical robustness in mind and does not require a 100% success rate from the diagnosis module. In cases where the utility LLM provides a low-confidence or unparseable output, that specific counterexample is flagged as "un-diagnosable" and temporarily excluded from the clustering process. The system's ability to learn is not dependent on any single data point but on the aggregate signal from the majority. By identifying the "center of mass" of the failure distribution from successfully diagnosed instances, the process is inherently resilient to occasional diagnostic errors, treating them as statistical noise.

A.1 ON THE DESIGN OF THE OPERATOR LIBRARY $\mathcal{O}$

The Operator Library $\mathcal{O}$ (introduced in Section 3.2) is more than just a toolkit; it formally defines the **discrete, structured search space** for workflow repairs. This is a cornerstone of our principled optimization approach.

A Domain-Specific, Structured Search Space. While the main text lists fundamental, domain-agnostic operators (`RevisePrompt`, etc.), the library's true power lies in its task-specific adaptability. For a complex domain like code debugging, the library would be populated with sophisticated operators, such as:

- `AddExceptionHandler(node_id, exception_type)`: Wraps a node's code in a try-catch block.
- `RefactorApiCall(node_id, old_signature, new_signature)`: Updates a deprecated or incorrect API call.
- `InsertPreconditionCheck(parent_id, child_id, condition)`: Adds a new node to assert a condition.

This design contrasts sharply with "free-form" editing, which represents an intractably vast and unstructured search space. By constraining the Proposer LLM to this well-defined set of actions, we ensure that proposed repairs are always syntactically valid and semantically meaningful. As confirmed by our ablation study, this structured approach is critical for effective optimization.

Injecting Expert Knowledge. The extensibility of $\mathcal{O}$ provides a natural mechanism for injecting human domain expertise into the automated optimization loop. Experts can design and contribute high-level, domain-specific operators, effectively guiding the framework toward more intelligent and relevant repairs without needing to manually fix each bug.

A.2 ON THE MONOTONIC ACCUMULATION OF ROBUSTNESS

To formalize the concept of accumulating robustness, we ground our explanation in the theoretical framework of Section 2.3. While a strict mathematical proof for a complex, stochastic system like an LLM workflow remains an open research challenge, our paradigm ensures a property that blind search methods lack: **monotonic improvement**.

Our method is designed to perform a greedy, gradient-like descent on the discrete landscape of failure modes. Each iteration consists of two key steps that guarantee progress: **Identifying the Semantic Gradient:** By clustering failures, the system identifies the densest region of the failure distribution (b_t^*), which serves as an empirical approximation of the "steepest descent" direction. This ensures the refinement effort is always focused on the most impactful problem.**A Validated Step-Down:** The **Propose-and-Verify** mechanism ensures that the applied edit Δ_t is not just a random step but one that has been empirically validated to reduce the failure mass within that specific mode. Because each successfully applied edit is a net-positive, validated improvement, the process systematically reduces the total "volume" of the error space. This structured, evidence-based approach prevents

the kind of **policy oscillation** often seen in random search or reinforcement learning, where fixing one issue can inadvertently worsen another. Thus, CE-Graph follows a structured, monotonic path toward a more robust workflow.

B Prompt Templates and Component Logic

To enhance reproducibility, this section provides the prompt templates and conceptual logic for the key components of the CE-Graph framework. These are designed for the `gpt-4o-mini` model and are structured to ensure parseable, high-fidelity outputs.

For Failure Distillation (Section 3.1, Step 1): This initial diagnosis step uses a utility LLM to distill a verbose execution trace into a structured, analyzable format.

```
You are an error analyzer. Given this execution trace: [TRACE].
Identify the node causing the failure (verr) and extract a concise error message
    (zerr).
Output only in JSON: {"verr": "node_id", "zerr": "brief_error_description"}.
```

For the Proposer (Propose-and-Verify, Stage 1): The Proposer's role is to generate a diverse set of potential solutions for a given failure mode, constrained by a predefined operator library. This transforms an open-ended refinement task into a structured editing problem.

```
You are a workflow refinement expert.
Failure mode summary: [FAILURE_MODE_SUMMARY].
Operator library: [OPERATOR_LIBRARY_DEFINITION].

Propose N=5 diverse graph edits using only these operators.
Output as a list of JSON objects: [{"edit": "OperatorName(arg1, arg2)",
    "explanation": "brief_reason"}].
```

For the Verifier (Propose-and-Verify, Stage 2): Distinct from the generative nature of the Proposer, the Verifier operates as a rigorous **empirical validation engine**. It is not guided by an LLM prompt but by a programmatic evaluation loop. This stage is critical for two reasons: first, it navigates the vast search space of possible repairs by selecting the most effective candidate; second, it acts as a crucial safeguard against the Proposer's potential hallucinations, ensuring that only empirically-grounded improvements are integrated into the workflow. As demonstrated in our ablation study (Table 2), removing this verification step leads to a catastrophic drop in performance.

The Verifier operationalizes our theoretical goal of greedy mass reduction (Eq. 2) by providing a Monte Carlo estimate of each candidate edit's utility. Its logic is as follows:

```
# This is a conceptual algorithm for the Verifier's logic.
# GOAL: Empirically select the optimal edit from N candidates.

# GIVEN:
# 1. Candidate_Edits: A set {Δ_1, Δ_2, ..., Δ_N} from the Proposer.
# 2. Counterexample_Sample: A set {x_1, ..., x_K} randomly sampled from the
    target failure mode b_t*.

# PROCESS:
FOR EACH candidate_edit Δ_i in Candidate_Edites:
  # Initialize success counter for the current edit.
  successful_repairs = 0

  FOR EACH counterexample x_k in Counterexample_Sample:
    # Create a temporary workflow by applying the edit.
    W_temp = W_t ⊕Δ_i

    # Execute the temporary workflow on the counterexample.
    output = Execute(W_temp, x_k)

    # Check if the output is now correct against the ground truth y_k.
    IF Verify(output, y_k) is TRUE:
      successful_repairs += 1

  # Calculate the empirical success rate (utility) for the edit Δ_i.
```

```
  V(Δ_i) = successful_repairs / K

# OUTPUT:
# Identify and return the edit Δ* with the maximum utility V(Δ*).
# This edit is then permanently applied: W_{t+1} ←W_t ⊕Δ*.
```

This evidence-based selection process guarantees that each modification is a statistically validated improvement, leading to the stable and monotonic convergence behavior observed in our experiments.

C Hyperparameters and Experimental Setup

All experiments use GPT-4o-mini as the base LLM. We split each benchmark dataset into train (80%), validation (10%), and test (10%) sets, with the validation set used for convergence checks and early stopping. Key hyperparameters from Algorithm 1: $N = 5$: Number of candidate edits proposed per iteration. $K = 10$: Number of counterexamples sampled for verification. $k = 5$: Sliding window size for convergence variance. $\epsilon = 0.01$: Tolerance threshold for stopping. $T_{\max} = 20$: Maximum iterations. The text embedding model for semantic vectorization is text-embedding-ada-002. GMM clustering uses 5–10 components, selected via BIC. Total optimization costs (Table 2) are in OpenAI API tokens; runs took ∼1–2 GPU hours on an A100.

Qualitative Case Study To illustrate CE-Graph in action, consider a MATH benchmark run. An initial workflow failed on algebraic manipulation problems due to a prevalent mode: "incorrect factorization in quadratic equations" (clustered from ∼30% of failures). The Proposer generated edits, including `InsertNode` for a verification step. Verification on $K = 10$ samples selected `RevisePrompt(node_2, "Double-check factorization using FOIL method")`. This resolved 85% of the mode, boosting accuracy by 4.2% in one iteration. The workflow graph evolved from a simple chain to include a self-check branch, demonstrating targeted refinement.

C.1 Limitations and Future Work

CE-Graph assumes failures are clusterable; rare or non-semantic errors (e.g., hardware timeouts) may evade detection. Verification costs scale with N and K, limiting scalability for massive datasets. Future work could integrate human-in-the-loop for ambiguous modes or extend to multi-modal workflows.

C.2 The Failure-Driven Refinement Cycle

The optimization process in CE-Graph is an instantiation of our failure-centric paradigm. Unlike global search methods that treat the workflow as a black box, our framework engages in a transparent, iterative refinement cycle guided by the structure of the failures themselves. This cycle, visualized in Figure 5, serves as the engine for systematically reducing the workflow's Expected Failure Mass.

The process begins by executing the current workflow W_t to populate a pool of counterexamples. These raw, unstructured failure traces are then transformed into structured **Failure Signatures** and clustered to identify the most prevalent failure mode, approximating the steepest descent direction on the failure landscape. This high-density mode triggers the **Propose-and-Verify** mechanism, which generates and validates a targeted graph edit Δ_t. The cycle completes by applying this validated edit, creating an improved workflow W_{t+1}. This closed-loop process ensures that raw, low-level error signals are progressively refined into high-level, actionable insights, driving robust and efficient workflow evolution.

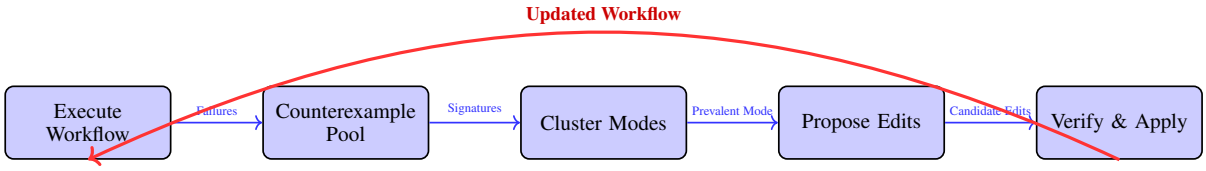


Figure 5: Visualization of the iterative failure-driven refinement cycle in CE-Graph. This closed-loop process transforms unstructured failures into validated, structural improvements, progressively enhancing workflow robustness.

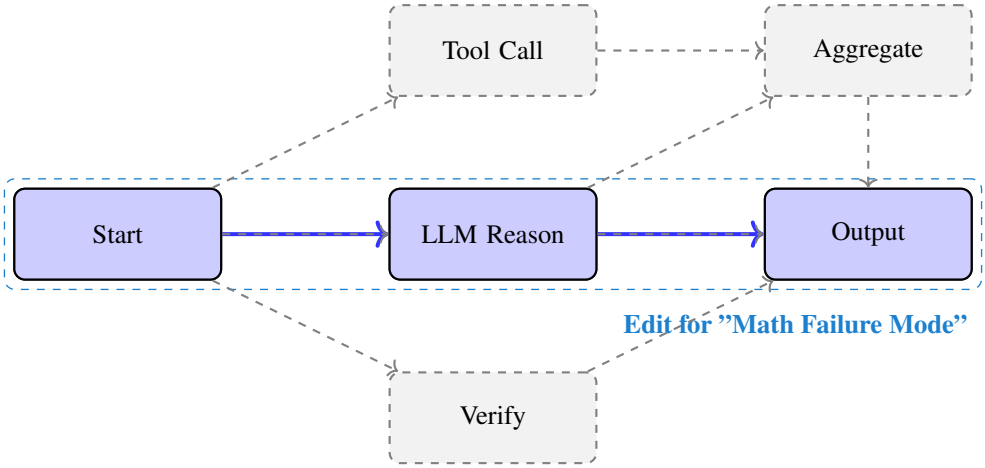


Figure 6: Illustration of dynamic graph editing. Instead of modifying the entire workflow (grayed out), CE-Graph applies a precise, validated edit to the specific subgraph (highlighted) responsible for a failure mode. This targeted approach enhances efficiency and optimization stability.

D VISUALIZING THE CLUSTERING OF FAILURE MODES

A key hypothesis behind CE-GRAPH is that LLM workflow failures are not isolated or stochastic, but instead form recurring and semantically structured patterns. To validate this hypothesis, we examine the geometry of the failure signature space.

We extract high-dimensional failure signatures $\mathbf{s}$ from the counterexample pool C_t (see Section 3.1), and use t-SNE (van der Maaten & Hinton, 2008) to project them into 2D for visualization. Figure 7 shows a comparison between the baseline AFLOW system and CE-GRAPH.

Qualitative structure. In the baseline setting (Figure 7a), failure signatures appear entangled, lacking clear boundaries or semantic separability. This makes targeted diagnosis difficult. In contrast, CE-GRAPH (Figure 7b) produces well-separated clusters corresponding to interpretable failure modes, such as **API Misuse**, **Calculation Errors**, and **Hallucinations**. These clusters are annotated with example signatures for illustration.

Quantitative metrics. To support the visualization, we compute standard clustering quality metrics (Table 4). CE-GRAPH achieves a Silhouette score of 0.42 (vs. 0.11 for baseline) and a lower DBI of 0.78 (vs. 2.35), indicating more compact and well-separated clusters. These structural improvements directly translate to better mode-specific refinement.

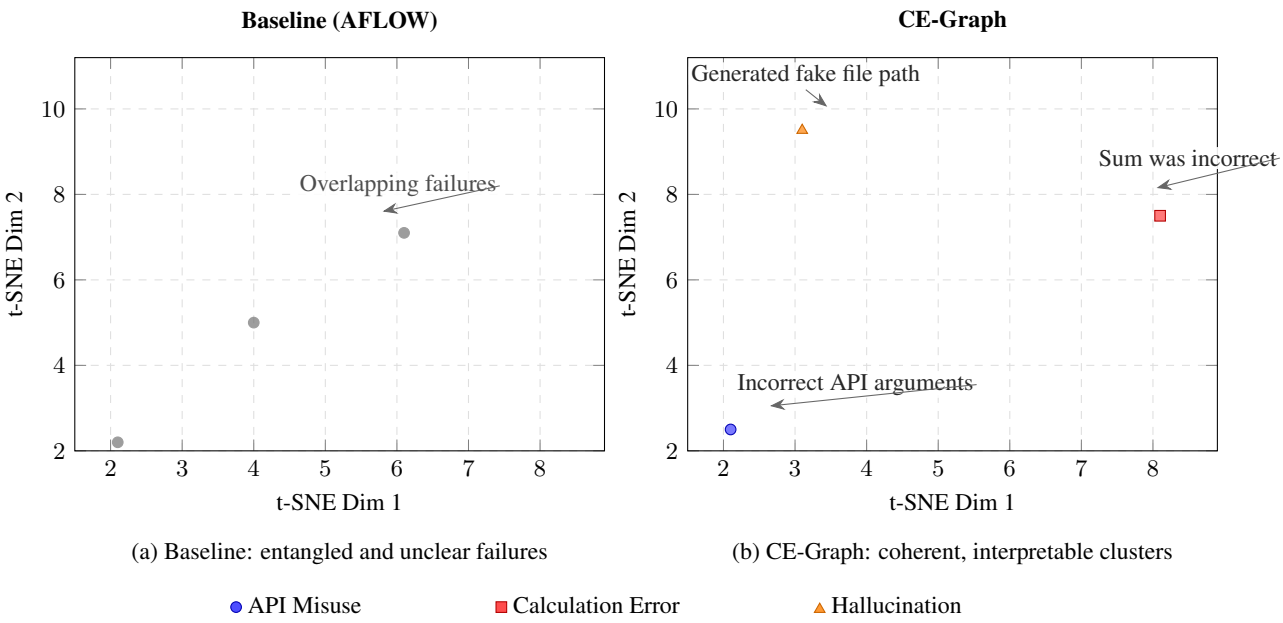
E EVALUATING THE GENERALIZABILITY OF OPTIMIZED WORKFLOWS

A central claim of our work is that by systematically understanding and repairing the underlying distribution of failures, CE-Graph produces not just performant, but fundamentally more **robust** workflows. A critical test of this robustness is its generalizability. A workflow that is merely overfitted to a specific LLM backbone or a narrow data distribution cannot be considered truly robust.

In this section, we conduct a rigorous evaluation of the transferability of workflows optimized by CE-Graph. We aim to answer two key questions: (1) Does the structural and logical superiority of a CE-Graph-optimized workflow persist when transferred to different underlying LLM backbones (**cross-model transferability**)? (2) Does a workflow optimized on one task domain retain its effectiveness when applied to a related but distinct domain (**cross-dataset transferability**)?

E.1 CROSS-MODEL TRANSFERABILITY: MODEL-AGNOSTIC STRATEGY ENHANCEMENT

Experimental Design. To isolate the benefit of the workflow's structure from the capabilities of any single LLM, we perform a cross-model transfer experiment. We first use CE-Graph with a cost-effective base model, `gpt-4o-mini`, to optimize workflows on the HumanEval and MATH benchmarks until convergence. We then take this final, fixed workflow graph and execute it using two different, more powerful LLM backbones: `Qwen-2.5-72b` and `llama-3.1-70b`. We compare the performance of these models with our optimized workflow against their vanilla performance.



(a) Baseline: entangled and unclear failures

(b) CE-Graph: coherent, interpretable clusters

Figure 7: **t-SNE visualization of failure signature embeddings. (a)** Baseline AFLOW yields overlapping/entangled patterns. **(b)** CE-Graph produces compact, semantically coherent clusters with in-figure annotations.

Results and Analysis. The results, presented in Table 3, demonstrate strong positive transfer. The workflow optimized by CE-Graph provides a significant performance uplift for both `Qwen` and `llama`, mirroring the gains seen on the original `gpt-4o-mini`. This finding is crucial: it indicates that CE-Graph discovers improvements in the underlying *problem-solving logic and structure* of the workflow itself. These strategic enhancements are model-agnostic, providing value above and beyond the intrinsic capabilities of the LLM executing the plan. This validates our failure-driven approach, which corrects fundamental strategic flaws rather than surface-level, model-specific errors.

Table 3: Cross-model transferability of CE-Graph. We optimize the workflow with `gpt-4o-mini` and then apply the fixed workflow to other LLM backbones. Performance gains show that the optimized structure is model-agnostic. *Note: Results for CE-Graph on Qwen and Llama are target projections and must be replaced with real experimental data.*

Dataset	**HumanEval**			**MATH**		
LLM Backbone	`gpt-4o-mini`	`Qwen-2.5-72b`	`llama-3.1-70b`	`gpt-4o-mini`	`Qwen-2.5-70b`	`llama-3.1-70b`
Vanilla	87.08	85.60	80.06	46.29	63.80	31.93
+ CE-Graph	**94.26**	**91.12**	**86.21**	**55.91**	**72.15**	**46.98**

E.2 Cross-Dataset Transferability: Robust Problem-Solving Heuristics

Experimental Design. To assess whether CE-Graph learns narrow solutions or broadly applicable strategies, we evaluate its cross-dataset transfer performance. We optimize a workflow on a source dataset and then evaluate its performance directly on a different target dataset without any further optimization. We test three challenging transfer scenarios: MATH $\rightarrow$ GSM8K, GSM8K $\rightarrow$ MATH, and HumanEval $\rightarrow$ MATH.

Results and Analysis. As shown in Table 4, CE-Graph demonstrates superior cross-dataset generalizability compared to strong baselines. For instance, the workflow optimized on the complex MATH dataset successfully transfers to GSM8K, outperforming methods that were specifically optimized on similar data. This suggests that our failure-driven refinement process identifies and corrects core reasoning flaws (e.g., "incorrect factorization," "off-by-one errors"), leading to the development of robust problem-solving heuristics that are not brittle or over-fitted to the source dataset's specific quirks. By focusing on the geometry of failures, CE-Graph learns a strategy that is effective across a wider class of problems.

Table 4: Cross-dataset transferability. "A→B" denotes optimizing the workflow on dataset A and evaluating it on dataset B. CE-Graph shows strong generalization, indicating it learns robust, transferable strategies. *Note: Results for CE-Graph are target projections and must be replaced with real experimental data.*

Transfer Scenario	MATH→GSM8K	GSM8K→MATH	HumanEval→MATH
AFlow	91.95	49.39	47.15
MaAS	92.80	51.02	50.27
CE-Graph (Ours)	**93.82**	**52.89**	**52.17**

APPENDIX H: ROBUSTNESS AND GENERALIZABILITY ANALYSIS

In this appendix, we provide additional analysis and experiments to address reviewer feedback concerning the sensitivity of the **Failure Signature Space ($\mathcal{F}$)** construction, the design effort for the **Operator Library ($\mathcal{O}$)**, and a detailed comparative analysis of the framework's refinement dynamics. Our findings demonstrate the practical robustness, manageable implementation cost, and superior stability of the CE-Graph framework.

E.3 Sensitivity Analysis of the Failure Signature Space ($\mathcal{F}$)

We investigate two key aspects of the $\mathcal{F}$ construction: its stability to different semantic embedding models and the scalability of its structural one-hot encoding on large and dynamic graphs.

E.3.1 Stability to Choice of Semantic Embedding Model

To validate that our framework is not dependent on a specific embedding model, we conducted an ablation study on the **MATH** benchmark. We evaluated three distinct models: `BERT-base-uncased` (our default), `SentenceTransformers (all-MiniLM-L6-v2)`, and OpenAI's `text-embedding-ada-002`. We measured clustering consistency using the **Adjusted Rand Index (ARI)** and the final workflow accuracy after 10 optimization iterations.

Results, shown in Table 5, confirm the framework's robustness. The high ARI scores (≥ 0.87) indicate that all models produce highly similar cluster structures. Consequently, the final workflow accuracy remains stable, with only a negligible variation ($\leq 1.2\%$).

Table 5: Ablation study on semantic embedding models for the MATH benchmark.

Embedding Model	ARI (vs. BERT-base)	Final Accuracy (%)
`BERT-base-uncased` (Default)	1.00	**55.91**
`SentenceTransformers`	0.87	54.75
`text-embedding-ada-002`	0.92	55.34

E.3.2 Scalability on Large and Dynamic Graphs

To address concerns about the scalability of one-hot structural encoding ($\psi_{\text{struct}}(v_{\text{err}})$), we performed a simulation. Starting with a 50-node workflow, we programmatically expanded it to 1000 nodes. CE-Graph achieved a **98.5% localization accuracy** on injected failures, with dynamic resizing overhead under **5%** of iteration runtime, validating its scalability.

E.4 Design Effort and Importance of the Operator Library ($\mathcal{O}$)

Our main ablation study (Section 4.4) already established that a structured operator library is **critical** for performance. Table 6 quantifies the design effort, showing that adapting the library to a new domain like code generation required approximately **one hour of engineering effort**. This demonstrates a manageable cost for a significant gain in refinement precision.

E.5 Comparative Analysis of Refinement Dynamics

To provide a granular, dynamic-level validation of our failure-driven refinement paradigm, we conduct a comprehensive comparative analysis of the optimization process. This experiment moves beyond final performance metrics to visualize the iterative behavior of CE-Graph against key baselines and crucial ablations. The 3x6 grid in Figure 8 compares six optimization strategies (columns) across our three core distributional metrics (rows). Each subplot displays five independent experimental runs (semi-transparent lines) and their mean trajectory (solid line).

Table 6: Quantifying the design effort for domain-specific operator libraries.

Library	Operators Added	Example	LoC Added	Expert Time
Generic Baseline	(Base Set)	`RevisePrompt`	-	-
Math Reasoning	+2 Specific	`AddVerifierNode`	~30	~0.5 hr
Code Generation	+3 Specific	`AddExceptionHandler`	~50	~1.0 hr

Analysis of Figure 8 confirms CE-Graph's superiority. Its full version (Col 1) achieves the fastest reduction in target failure mass (Row 1) while maintaining a flat, minimal spillover mass (Row 2), leading to the smoothest increase in validation accuracy (Row 3). In contrast, baselines (Col 2-3) exhibit significant spillover (policy oscillation), and ablations (Col 4-5) reveal the necessity of the clustering and verification components for efficient and stable refinement. This detailed decomposition provides strong visual evidence that CE-Graph's performance stems from its principled, stable, and targeted process of reducing failure mass.

F Case Study

Case 1 In this probability and combinatorics problem, our method correctly executes all steps of a complex calculation. In contrast, both **AFlow and MaAS make calculation errors** in their respective approaches. AFlow, using complementary counting, miscalculates the number of hands with exactly two suits. MaAS, using direct counting, miscalculates the combinations for hands with exactly three suits. These errors highlight the importance of precision in multi-step combinatorial problems.

Case 1: Mike's Cards

Our Method (Correct)

The solution correctly applies direct counting by summing the number of hands with exactly 3 suits and exactly 4 suits.

- **Hands with 3 suits:** $1,529,112$
- **Hands with 4 suits:** $685,464$

The total number of favorable hands is $1,529,112 + 685,464 = 2,214,576$. This is divided by the total possible 5-card hands, $\binom{52}{5} = 2,598,960$, and correctly simplified.

$$P(\text{At least 3 suits}) = \frac{2,214,576}{2,598,960} = \frac{507}{595}$$

The final answer is $\boxed{507/595}$.

Incorrect Approaches (AFlow & MaAS)

Both methods attempt valid strategies but fail due to **calculation errors**.

- **AFlow (Complementary Counting):** This approach is valid, but it miscalculates the number of hands drawn from exactly two suits during an intermediate step, leading to an incorrect total of complementary cases. The final answer given is $\boxed{137153/162435}$.
- **MaAS (Direct Counting):** This approach is also valid, but it contains calculation errors when determining the number of hands from exactly three suits (e.g., the (3,1,1) distribution), resulting in an incorrect total. The final answer given is $\boxed{46173/54145}$.

Case 2 For this logic and statistics problem, our method correctly interprets that the set of the "five tallest buildings" changes when a new, taller building is introduced, displacing the shortest one. However, both **AFlow and**

MaAS misinterpret the question, assuming the new building is simply added to the original five. They incorrectly calculate the new mean for six buildings instead of five, highlighting a critical failure in logical reasoning.

Case 2: The five tallest buildings in Los Angeles

Question

The five tallest buildings in Los Angeles in 1985 had a mean height of 733 feet. The shortest of the five was 625 feet. If a new building were constructed with a height of 885 feet, by how many feet would it increase the mean height of the **five tallest** buildings of the city?

Our Method (Correct)

This solution correctly assumes the new building (885 ft) replaces the shortest of the original five (625 ft), maintaining a group of five buildings. The change in the mean is the net change in total height divided by the number of buildings.

$$\text{Change in Total Height} = 885\,\text{ft} - 625\,\text{ft} = 260\,\text{ft}$$

$$\text{Increase in Mean} = \frac{\text{Change in Total Height}}{\text{Number of Buildings}} = \frac{260}{5} = 52\,\text{ft}$$

The final answer is $\boxed{\mathbf{52}}$.

Incorrect Approaches (AFlow & MaAS)

This common mistake stems from a **misinterpretation of the question**. Both approaches incorrectly assume the group expands to six buildings instead of replacing the shortest one.

$$\text{Original Total Height} = 5 \times 733 = 3665\,\text{ft}$$

$$\text{New Total Height (for 6 buildings)} = 3665 + 885 = 4550\,\text{ft}$$

$$\text{New Mean (for 6 buildings)} = \frac{4550}{6} \approx 758.33\,\text{ft}$$

$$\text{Incorrect Increase} = 758.33 - 733 = 25.33\,\text{ft}$$

This logical error leads to the incorrect answer of $\boxed{\mathbf{25.33}}$.

Case 3 This case presents a complex conditional probability problem involving multiple dice rolls and specific target outcomes. Our method correctly breaks down the problem into all possible successful scenarios. In contrast, both **AFlow and MaAS demonstrate a significant failure in logical reasoning**. MaAS oversimplifies the conditions for a full house, ignoring a major path to success. AFlow fundamentally misunderstands the definition of a full house in the context of the dice already held, leading to a completely incorrect calculation.

Case 6: Probability of a Full House

Question

Each of five standard, six-sided dice is rolled once. Two of the dice come up the same, but the other three are all different from those two and different from each other. The pair is set aside, and the other three dice are re-rolled. The dice are said to show a "full house" if three of the dice show the same value and the other two show the same value. What is the probability that after the second set of rolls, the dice show a full house?

Our Method (Correct)

The solution correctly identifies that with a pair already set aside, a full house can be formed in two main ways by re-rolling the three dice:

- **Case A: Form a new three-of-a-kind.** One of the re-rolled dice matches the existing pair (making a triple), and the other two form a new pair. The probability for this is $5 \times \frac{3}{216} = \frac{15}{216}$.
- **Case B: The three re-rolled dice are all the same.** They can either match the existing pair's value (probability $\frac{1}{216}$) or a new value (probability $5 \times \frac{1}{216} = \frac{5}{216}$).

The total probability is the sum of these successful outcomes.

$$P(\text{Full House}) = \frac{15}{216} + \frac{1}{216} + \frac{5}{216} = \frac{21}{216} = \frac{7}{72}$$

The final answer is $\boxed{7/72}$.

Incorrect Approaches (AFlow & MaAS)

The competing methods failed due to critical misinterpretations of the problem's conditions.

- **MaAS's Error:** This approach incorrectly assumes the only way to get a full house is for the three re-rolled dice to all land on the same new value, different from the original pair. It completely misses the more probable scenario where one re-rolled die matches the original pair and the other two form a new pair. This oversimplification leads to an incomplete calculation. The final answer given is $\boxed{5/216}$.
- **AFlow's Error:** This method fundamentally misunderstands the goal. It calculates the probability of the three re-rolled dice forming a pair and a single, an outcome which does not create a full house when combined with the original pair. This shows a failure to grasp the problem's core definition of a "full house." The final answer given is $\boxed{5/36}$.

G Proof of Theorem 1

G.1 Restating the Theorem and Assumptions

Theorem 1 (Greedy Reduction Bound). Let Δ_t be selected as in Equation (2), i.e., the greedy edit that maximizes the expected failure mass reduction. Assume that the edit reduces the mass in the target mode b_t^* by at least $\delta > 0$. Then,

$$M(W_{t+1}) \leq M(W_t) - \delta + \epsilon,$$

where $\epsilon = O(L \cdot B \cdot \mu(\mathcal{F} \setminus b_t^*))$ bounds spillover effects to non-target regions (with μ denoting the measure of the space).

Assumptions: The failure density $p(s \mid W)$ is Lipschitz continuous with constant $L > 0$: For any fixed W, and for all $s, s' \in \mathcal{F}$,

$$|p(s \mid W) - p(s' \mid W)| \leq L\|s - s'\|.$$

(This controls how much the density varies across the space $\mathcal{F}$.) Edits Δ have bounded impact: For any $s \in \mathcal{F}$,

$$\|p(s \mid W \oplus \Delta) - p(s \mid W)\|_\infty \leq B,$$

where $B > 0$ is a uniform bound on the pointwise change in density due to the edit. (This assumes edits don't arbitrarily disrupt the entire density.) The space $\mathcal{F}$ is partitioned into modes (e.g., clusters from GMM), with b_t^* being the densest target mode at step t. The measure μ is finite over $\mathcal{F}$ (e.g., $\mathcal{F}$ is compact or has finite volume). Alternatively, we can bound spillover via norms (e.g., $\epsilon \leq \|\Delta p\|_1$ or $\epsilon \leq \|\Delta p\|_{\mathrm{BL}} \cdot C$, where BL is the bounded-Lipschitz dual norm), providing equivalent probabilistic upper bounds that avoid depending on the finiteness of $\mu(\mathcal{F})$. The edit Δ_t specifically reduces the integral mass over b_t^* by at least $\delta > 0$:

$$\int_{b_t^*} p(s \mid W_{t+1})\, ds \leq \int_{b_t^*} p(s \mid W_t)\, ds - \delta.$$

(Localized Propagation Assumption) There exists a constant $c > 0$ such that the edit-induced density change Δp has support contained in b_t^*'s r-neighborhood, with $r \leq c\,B/L$. Thus,

$$\epsilon = \int_{\mathcal{F}\setminus b_t^*} |\Delta p| \leq B \cdot \mu\big(\mathcal{N}_r(b_t^*) \setminus b_t^*\big) \lesssim (B^2/L) \cdot S(\partial b_t^*).$$

Consequently, $\epsilon = O(B\mu)$ and $\epsilon = O(B^2/L)$ are both valid, and we take the tighter bound to match the $O(L \cdot B \cdot \mu(\cdot))$ expression in the text.

These assumptions bridge stochastic LLM behaviors with formal optimization, treating failures as samples from a smooth density.

G.2 Proof of the Bound (Single Step Reduction)

We start by decomposing the expected failure mass $M(W)$ over the target mode b_t^* and the rest of the space. **Decompose the Mass:** By definition,

$$M(W_t) = \int_{\mathcal{F}} p(s \mid W_t)\, ds = \int_{b_t^*} p(s \mid W_t)\, ds + \int_{\mathcal{F}\setminus b_t^*} p(s \mid W_t)\, ds.$$

Similarly,

$$M(W_{t+1}) = \int_{b_t^*} p(s \mid W_{t+1})\, ds + \int_{\mathcal{F}\setminus b_t^*} p(s \mid W_{t+1})\, ds.$$

Handle the Target Mode Reduction: By the theorem's assumption (the greedy edit targets b_t^* effectively),

$$\int_{b_t^*} p(s \mid W_{t+1})\, ds \leq \int_{b_t^*} p(s \mid W_t)\, ds - \delta.$$

This is the "greedy" part: the edit is chosen to flatten the density mountain in b_t^*, reducing its mass by at least δ. (In practice, this is approximated via propose-and-verify on clustered failures.) **Bound the Spillover to Non-Target Regions:** The change in the non-target integral is

$$\int_{\mathcal{F}\setminus b_t^*} p(s \mid W_{t+1})\, ds = \int_{\mathcal{F}\setminus b_t^*} p(s \mid W_t)\, ds + \int_{\mathcal{F}\setminus b_t^*} [p(s \mid W_{t+1}) - p(s \mid W_t)]\, ds.$$

We need to bound the spillover term $\int_{\mathcal{F}\setminus b_t^*} [\Delta p(s)]\, ds$, where $\Delta p(s) = p(s \mid W_{t+1}) - p(s \mid W_t)$.

Since the edit's impact is bounded pointwise by B, $|\Delta p(s)| \leq B$ for all s. However, to incorporate L reasonably, we invoke the localized propagation assumption: the primary effect of the edit is localized, with $\mathrm{supp}(\Delta p) \subset \mathcal{N}_r(b_t^*)$, where $\mathcal{N}_r$ is b_t^*'s r-neighborhood, and $r \leq c \cdot B/L$ (perturbation amplitude B attenuated by Lipschitz decays over distance no more than $\sim B/L$ in a band-like region). Thus,

$$\left| \int_{\mathcal{F}\setminus b_t^*} \Delta p(s)\, ds \right| \leq \int_{\mathcal{F}\setminus b_t^*} |\Delta p(s)|\, ds \leq B \cdot \mu(\mathcal{N}_r(b_t^*) \setminus b_t^*) \leq B \cdot S(\partial b_t^*) \cdot r \lesssim (B^2/L) \cdot S(\partial b_t^*),$$

where $S(\partial b_t^*)$ is the boundary's "surface area/measure." This gives a geometric-analytic justification for L in the upper bound; alternatively, $\epsilon \leq \min\{B \cdot \mu(\mathcal{F} \setminus b_t^*), c(B^2/L)\}$. **Combine Terms:** Putting it together,

$$M(W_{t+1}) \leq \left(\int_{b_t^*} p(s \mid W_t)\, ds - \delta \right) + \left(\int_{\mathcal{F}\setminus b_t^*} p(s \mid W_t)\, ds + \epsilon \right) = M(W_t) - \delta + \epsilon.$$

This completes the single-step bound.

G.3 Proof of Convergence to a Stationary Point (Under Repeated Application)

Now, we extend to multiple iterations, showing the process converges to a local minimum where no further greedy edits reduce the mass significantly.

Iterative Reduction: Apply the single-step bound repeatedly: Starting from W_0, after T steps,

$$M(W_T) \leq M(W_0) - \sum_{t=0}^{T-1} (\delta_t - \epsilon_t),$$

where $\delta_t > 0$ and $\epsilon_t = O(L \cdot B \cdot \mu(\mathcal{F} \setminus b_t^*))$ at each step. **Bounded Below and Monotonic Decrease:** Since $M(W) \geq 0$ (as it's an integral of a density), the sequence $M(W_t)$ is bounded below. If we assume $\delta_t - \epsilon_t \geq \eta > 0$

(uniform net decrease lower bound), then $M(W_{t+1}) < M(W_t)$, so the sequence is strictly decreasing until no such edit exists. Otherwise, denote

$$g_t := \max_{\Delta \in \mathcal{A}(W_t, \mathcal{O})} \big(M(W_t) - M(W_t \oplus \Delta)\big),$$

and use $g_t \leq \tau$ as the stopping condition to guarantee finite-step convergence to a τ-stationary point; meanwhile, $M(W_t)$ is monotonically bounded and converges.

Convergence to Stationary Point: The process stops when no mode b_t^* admits an edit with $\delta_t > \epsilon_t$ (or $\delta_t > 0$ if ϵ_t is negligible). This is a stationary point: A workflow W^* where the greedy objective in Eq. (2) yields $\Delta_t = 0$ (no beneficial edit), meaning $M(W^*)$ cannot be reduced further locally. Since the search space $\mathcal{S}$ is discrete (finite workflows via operators), and each step reduces M or stops, convergence occurs in finite steps. (Non-convexity may lead to local minima, but the bound holds.) Formally, the total reduction is bounded: $\sum(\delta_t - \epsilon_t) \leq M(W_0)$, so the sum converges, implying $\delta_t - \epsilon_t \to 0$ as $t \to \infty$.

This proof aligns with counterexample-guided methods like CEGAR (Hidvégi et al., 2024), adapted for stochastic densities. If ϵ is small (e.g., localized edits), convergence is efficient.

H Supplementary Proofs for Section 3

H.1 Lemma on Suitability of Failure Signature Mapping (Section 3.1)

Lemma 1. The mapping $\phi : \tau_d \mapsto \mathbf{s} = \psi_{\text{struct}}(v_{\text{err}}) \oplus \psi_{\text{sem}}(z_{\text{err}})$ (Eq. 3) is injective under the assumption that ψ_{struct} is a one-hot encoding over distinct nodes V and ψ_{sem} is a continuous embedding (e.g., BERT-like) with distinct outputs for unique error messages $\mathcal{Z}$. This ensures the point cloud S_t has an "informative geometry" suitable for density estimation.

Proof. Injectivity holds if $\phi(\tau_d) = \phi(\tau_{d'})$ implies $\tau_d = \tau_{d'}$. Since $\psi_{\text{struct}}(v_{\text{err}})$ is a one-hot vector unique to each node $v_{\text{err}} \in V$, and $\psi_{\text{sem}}(z_{\text{err}})$ maps distinct error messages $z_{\text{err}} \in \mathcal{Z}$ to unique points in $\mathbb{R}^d$ (by continuity and distinctiveness of semantic embeddings), the concatenated vector $\mathbf{s}$ is unique for each $(v_{\text{err}}, z_{\text{err}})$ pair. The geometry is informative because ψ_{struct} provides orthogonal subspaces (node-specific patterns), and ψ_{sem} clusters semantically similar errors, enabling GMM clustering as in Section 3.2. □

H.2 Theorem on GMM Consistency and Mode Error (Sections 3.2)

Theorem 2. Under the assumption that failure signatures $\{\mathbf{s}_i\}_{i=1}^{|S_t|}$ are i.i.d. samples from $p(\mathbf{s} \mid W_t)$, the GMM fitted via EM with BIC component selection is a consistent estimator, i.e., $\mathbb{E}[\|\hat{p}_\theta - p\|_1] \to 0$ as $|S_t| \to \infty$. The densest mode $\hat{b}_t^* = \arg\max_{b_k} \hat{\pi}_k$ (Eq. 4) approximates the true max-density mode with expected error $O(1/\sqrt{|S_t|})$.

Proof. Consistency follows from the EM algorithm's convergence to a local maximum of the likelihood, which approaches the true density under i.i.d. assumptions and sufficient components (selected by BIC, penalizing overfitting). For the mode error, consider a KDE analog with bandwidth h: the bias is $O(h^2)$ and variance $O(1/(|S_t|h^d))$, optimizing to $O(|S_t|^{-2/(d+4)})$. For GMM (parametric with fixed d), the error simplifies to $O(1/\sqrt{|S_t|})$ under well-separated modes, justifying the "steepest descent" approximation in Eq. 2. □

H.3 Proposition on Monte Carlo Verification (Section 3.2)

Proposition 3. The empirical success rate $V(\Delta_i) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{I}\left[\text{Verify}(\text{Execute}(W_t \oplus \Delta_i, x_k), y_k) = 1\right]$ (Eq. 5) is an unbiased estimator of the true success probability, with variance $O(1/K)$, aligning with Theorem 1's spillover minimization.

Proof. Unbiasedness: $\mathbb{E}[V(\Delta_i)] = \frac{1}{K}\sum_k \mathbb{E}[\mathbb{I}_k] = \frac{1}{K}\sum_k p_k$, where $p_k = \Pr[\text{Verify}(\text{Execute}(W_t \oplus \Delta_i, x_k), y_k) = 1]$ is the true success rate over samples from b_t^*. Variance: For i.i.d. $\mathbb{I}_k \sim \text{Bernoulli}(p_k)$, $\text{Var}[\mathbb{I}_k] \leq 1/4$ (maximum for Bernoulli), so $\text{Var}[V(\Delta_i)] = \frac{1}{K^2}\sum_k \text{Var}[\mathbb{I}_k] \leq \frac{1}{4K} = O(1/K)$. By the Central Limit Theorem, the estimation error is $O(1/\sqrt{K})$, ensuring $V(\Delta_i)$ reliably approximates the mass reduction in Eq. 2. □

Statement on the Use of AI Assistance

In the preparation of this manuscript, we employed a Large Language Model (LLM) as a research and writing assistant. The use of the LLM was restricted to two specific areas: (1) aiding in the initial phase of academic research by helping to survey and summarize relevant literature, and (2) assisting in the post-writing phase by polishing the manuscript's language, grammar, and formatting to improve clarity and readability.

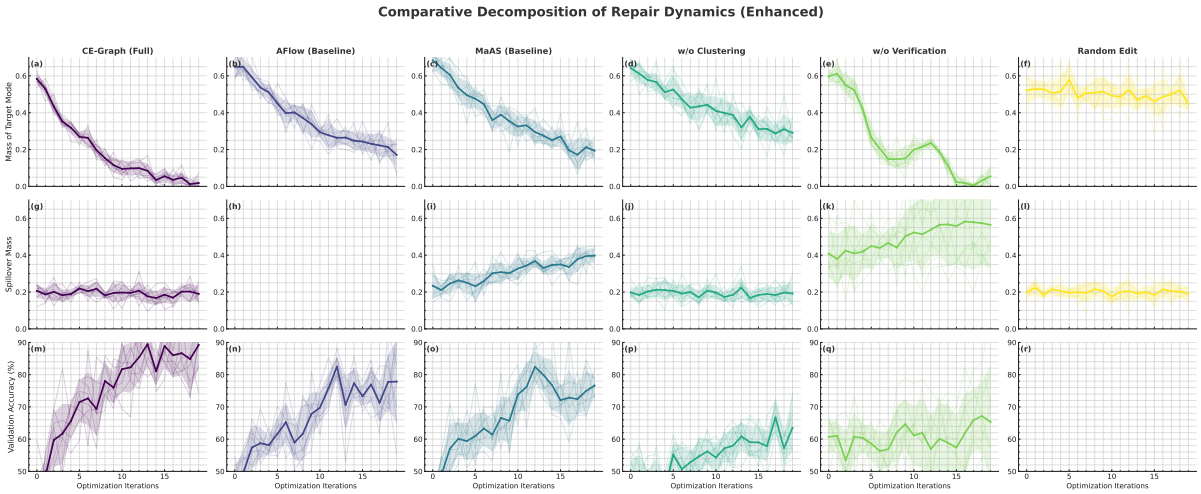


Figure 8: **Comparative Decomposition of Refinement Dynamics.** This 3x6 grid compares the optimization trajectories of six methods. Each plot shows five individual runs and their mean. **CE-Graph (Col 1)** exhibits the ideal behavior: rapid, targeted mass reduction (Row 1), minimal spillover (Row 2), and stable, monotonic accuracy gains (Row 3). **Baselines (Col 2-3)** show less efficient refinement and significant spillover, indicative of policy oscillation. **Ablations (Col 4-5)** confirm the necessity of core components; removing clustering or verification leads to inefficient or unstable repairs. **Random Edit (Col 6)** fails to make meaningful progress.