DDCG: Decoupled Dual-Critic Guidance for Embodied Agents

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have enabled embodied agents to perform high-level planning, yet grounding language-based plans into the physical world remains challenging. While feedback-based closed-loop planning is the dominant paradigm, existing methods typically rely on a single scalar feedback signal, which conflates physically infeasible actions that violate hard constraints with strategically suboptimal but valid choices. We identify this structural limitation as **Signal Confounding**, which leads to ambiguous feedback and unreliable plan correction. To address this issue, we propose **Decoupled Dual-Critic Guidance (DDCG)**, a framework that explicitly disentangles feedback into two complementary signals: a **Feasibility Critic** (C_F) that enforces physical and logical executability constraints, and a **Quality Critic** (C_Q) that evaluates the strategic value of feasible actions. This decoupled guidance enables precise error attribution during planning without requiring parameter updates to the LLM planner. We provide a theoretical interpretation of DDCG as critic-regularized planning with hard feasibility constraints. Experiments on two embodied planning benchmarks, VirtualHome and ScienceWorld, show that DDCG consistently improves plan executability and task success, particularly in long-horizon and generalization settings, highlighting the importance of structured feedback for reliable LLM-based planning.

1 Introduction

In recent years, the rapid advancement and widespread deployment of Large Language Models (LLMs) have unlocked new possibilities for embodied artificial intelligence. In particular, improvements in reasoning and planning capabilities have positioned LLMs as promising task planners, capable of decomposing high-level goals into executable action sequences for embodied agents. However, real-world embodied tasks are governed by complex physical constraints and logical dependencies, posing fundamental challenges for reliable task planning. Empirical evidence shows that even recent large-scale language models frequently generate plans that are physically infeasible, logically inconsistent, or factually incorrect (Huang et al., 2022a; Wei et al., 2024). These failures underscore the need for more reliable planning mechanisms for embodied agents (Fu et al., 2024; Chang et al., 2024).

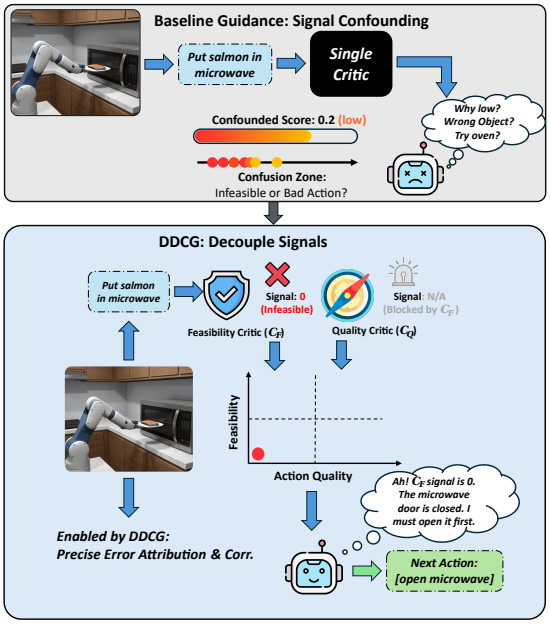

Figure 1: Compared to conventional closed-loop planning, DDCG disentangles error signals, enabling precise and effective replanning.

While feedback-based closed-loop planning has become the dominant paradigm for LLM-based embodied agents, we argue that existing approaches share a fundamental structural limitation. Specifically, they reduce a multi-dimensional evaluation over the state–action space—capturing feasibility, strategic quality, and long-horizon consequences—into a single, confounded feedback signal. This dimensionality collapse inevitably in-

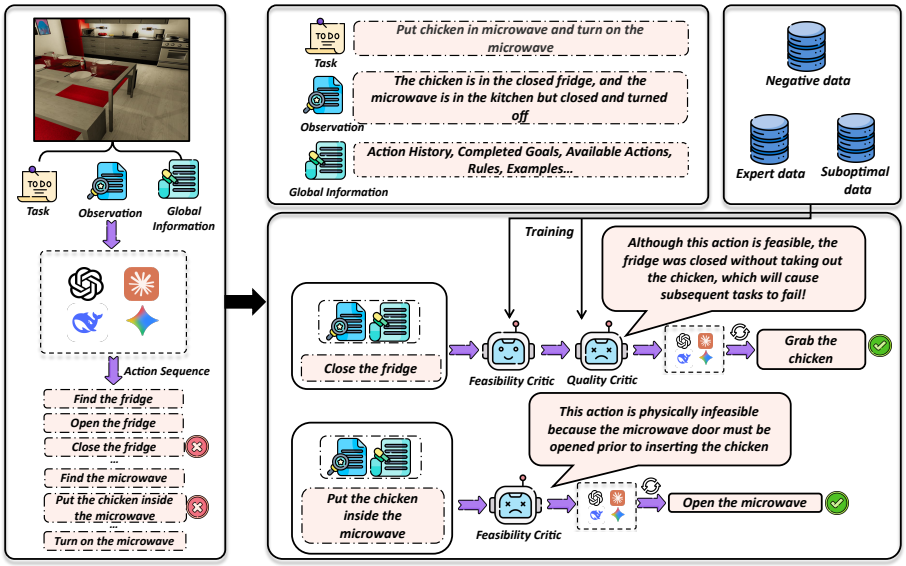

Figure 2: Overview of the **D**ecoupled **D**ual-**C**ritic **G**uidance (DDCG) pipeline. A base LLM planner proposes candidate actions, which are first filtered by a Feasibility Critic (C_F) to remove infeasible actions and then evaluated by a Quality Critic (C_Q) to select strategically effective ones, mitigating signal confounding during planning.

duces information loss: feasibility violations (hard constraint failures) and poor strategic choices (sub-optimal but valid actions) become indistinguishable to the planner. As a result, the agent cannot reliably attribute errors to their underlying causes, leading to inefficient or unstable plan refinement. We refer to this failure mode as **Signal Confounding**.

To address this issue, we propose **Decoupled Dual-Critic Guidance (DDCG)**, a framework that explicitly disentangles feedback signals by separating feasibility and strategic quality into two specialized critics. The **Feasibility Critic** (C_F) learns a model of the environment's hard physical and logical constraints, pruning the action space to a state-dependent safe set by enforcing constraint satisfaction. The **Quality Critic** (C_Q) learns a heuristic value function that evaluates the strategic merit of feasible actions with respect to task progress and resource efficiency. This decoupled design enables precise error attribution and targeted plan optimization.

From a theoretical perspective, DDCG can be viewed through the lens of constrained optimization. The Feasibility Critic plays a role analogous to a learned constraint function in a Constrained Markov Decision Process (CMDP) (Altman, 2021), defining a safe action subspace, while the Quality Critic provides dense utility guidance within this subspace.

Our contributions can be summarized as follows: (1) We identify and formally define Signal Confounding as a structural obstacle arising from collapsing multi-dimensional state–action evaluations into a single scalar feedback signal in LLM-based embodied planning. (2) We propose DDCG, a decoupled dual-critic framework that resolves this issue by independently modeling feasibility and strategic quality, enabling reliable error attribution and robust closed-loop planning. (3) We demonstrate through experiments on embodied planning benchmarks that DDCG consistently improves both plan executability and task success, particularly in long-horizon and generalization settings.

2 Related Work

LLM-based Embodied Planning. Recent advances in large language models have motivated their application to embodied planning and decision-making tasks, where agents decompose high-level goals into executable action sequences (Huang et al., 2022a; Chang et al., 2024). Prior work has explored both open-loop planning approaches, which generate full plans without environment interaction (Singh et al., 2023; Chen et al., 2021), and closed-loop methods that iteratively refine plans through feedback (Bai et al., 2024). Representative closed-loop paradigms include self-feedback and reflection-based methods, such as ReAct (Yao et al., 2023) and Inner Monologue (Huang et al., 2022b), as well as their extensions that incorporate memory or iterative self-correction (Shinn et al., 2023; Madaan et al., 2023;

Qu et al., 2024). Other approaches introduce external guidance via discriminators or auxiliary models to improve planning reliability (Qian et al., 2025; Chen et al., 2025). While these methods improve planning performance, they typically rely on a single feedback signal to guide action selection, leaving different failure modes implicitly conflated.

Guidance and Feedback for Decision-Making. Beyond embodied planning, a growing body of work studies how auxiliary feedback signals can guide LLM decision-making at inference time. Prior approaches have explored using learned reward models, value estimators, or discriminators to rank, filter, or critique candidate actions without updating the underlying language model (Ahn et al., 2022; Yang et al., 2025; Xia et al., 2025). Related studies in agent self-critique and verifier-based reasoning similarly rely on scalar feedback to steer generation (Madaan et al., 2023; Saunders et al., 2022). However, these approaches generally compress feasibility constraints and strategic utility into a single scalar score, which can obscure the source of errors during planning. In contrast, our work explicitly disentangles feasibility and quality signals, demonstrating that separating hard constraints from strategic evaluation enables more precise error attribution and more robust planning behavior in embodied environments.

3 Preliminary

CMDP formulation. We formulate embodied task planning as a Constrained Markov Decision Process (CMDP) (Altman, 2021), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, C, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the textual state and action spaces, $P(s'|s,a)$ the transition dynamics, $R(s,a)$ the reward function, $C(s,a)$ a cost function, and γ the discount factor. The objective is to learn a policy π that maximizes expected cumulative reward while satisfying a cost constraint.

In our setting, any action that violates physical or logical rules is considered infeasible and assigned a positive cost, while feasible actions incur zero cost. We enforce strict safety by setting the cost threshold to $d = 0$, which implies that a valid policy must assign zero probability to infeasible actions.

4 Method

This section presents DDCG, a framework for guiding LLM-based embodied planning via structured, decision-time feedback. As illustrated in Figure 2, DDCG augments a frozen LLM planner with two complementary critics that separately evaluate action feasibility and strategic quality. We first introduce the decoupled critic design (Section 4.1), with structured supervision described in Section 4.2, then describe how these critics are integrated into an online planning loop via in-context adaptation (Section 4.3), and finally provide a theoretical interpretation grounded in critic-regularized optimization with a policy improvement guarantee (Section 4.4).

4.1 Decoupled Dual-Critic Guidance

As discussed in Section 1, existing feedback mechanisms for LLM-based planners suffer from a fundamental structural limitation that we term **Signal Confounding**. This issue arises from collapsing a multi-dimensional evaluation over the state–action space—capturing feasibility and strategic quality—into a single scalar signal. As a result, physically infeasible actions and suboptimal but valid actions become indistinguishable, preventing precise error attribution and effective replanning.

To address this limitation, we propose **Decoupled Dual-Critic Guidance (DDCG)**, which explicitly factors feedback into two complementary components: a feasibility critic and a quality critic. By disentangling hard constraints from strategic utility, DDCG restores the structure lost under single-score guidance, as illustrated in Figure 1.

Feasibility Critic (C_F). The Feasibility Critic $C_F : \mathcal{S} \times \mathcal{A} \to \{0, 1\}$ models hard physical and logical constraints of the environment. Given a state s and a candidate action a, C_F predicts whether the action is executable. Actions that violate physical rules (e.g., placing an object without holding it) or logical preconditions (e.g., opening an already open container) are classified as infeasible. This effectively defines a state-dependent safe action set

$$\mathcal{A}_{\text{safe}}(s) = \{a \in \mathcal{A} \mid C_F(s,a) = 1\},$$

which prunes the action space to only those actions that satisfy hard constraints.

Quality Critic (C_Q). Conditioned on feasibility, the Quality Critic $C_Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ evaluates the strategic merit of feasible actions. It provides a dense, implicit reward signal $r_Q(s,a) = C_Q(s,a)$ that distinguishes among multiple valid options based on their expected contribution to task completion and efficiency. For example, while many

actions may be feasible in a given state, actions that advance the task goal receive higher quality scores. In this way, C_Q induces a smooth utility landscape over the safe action set, guiding planning beyond mere executability.

Importantly, neither critic alone is sufficient: C_F enforces hard safety constraints but cannot distinguish strategic quality, while C_Q provides utility guidance but relies on C_F to prevent invalid actions. Their decoupled integration is therefore essential for resolving Signal Confounding.

4.2 Data Generation via Structured Preference Signals

Recent advances in preference-based learning, such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) and Implicit Preference Optimization (IPO) (Garg et al., 2025), have demonstrated that learning from relative preferences can be more stable than relying on explicitly specified rewards. Inspired by this perspective, we design our data generation process to construct structured supervisory signals that encode preference information aligned with the decoupled critic design of DDCG.

Instead of learning a single scalar reward, we decompose preference supervision along two orthogonal dimensions—feasibility and strategic quality—and generate tailored signals for the Feasibility Critic C_F and the Quality Critic C_Q, respectively. This construction ensures that each critic learns a well-defined preference structure corresponding to its role.

Feasibility Supervision (C_F). The Feasibility Critic is trained to learn a sharp decision boundary over the safe action set $\mathcal{A}_{\text{safe}}(s)$. To provide unambiguous supervision, we construct binary preference signals using two types of data:

- **Expert actions**, which are inherently feasible and treated as positive examples.
- **Hard negative actions**, generated by applying rule-based perturbations that violate physical or logical preconditions (e.g., attempting to `put` an object that is not held). These samples explicitly encode infeasible actions and enforce hard environmental constraints.

Quality Supervision (C_Q). In contrast, strategic quality requires a dense, graded preference signal over feasible actions. We therefore construct a continuous quality spectrum with fixed anchor points:

Algorithm 1 DDCG Planning with Feasibility and Quality Checks

Require: LLM Planner, trained C_F, trained C_Q, threshold τ
  Given current state s and goal g
  while true **do**
    Propose candidate action a
      $\leftarrow$ LLM-PLANNER(s, g)
  Feasibility Check
    if $C_F(s, a) = 0$ **then**
      A feedback signal indicating infeasibility is delivered to the LLM
      continue
    end if
  Quality Check
    $score_{C_Q} \leftarrow C_Q(s, a)$
    if $score_{C_Q} \geq \tau$ **then**
      Execute action a in the environment
      break
    else
      A feedback signal indicating low quality is delivered to the LLM
    end if
  end while

- **Intermediate-quality actions**, obtained by generating multiple candidate actions for a given state and ranking them by semantic similarity to a high-quality reference action. These actions provide relative preference supervision within the feasible set.
- **Anchor actions**, where expert actions define the upper bound of the quality scale and hard negative actions define the lower bound.

This structured construction yields complementary preference signals: C_F enforces hard safety constraints by filtering infeasible actions, while C_Q induces a smooth utility landscape over feasible actions for strategic guidance. Together, these signals directly support the decoupled feedback mechanism of DDCG. Additional details on data distributions are provided in Section C.4.

4.3 Online Planning with Decoupled Guidance

DDCG guides a frozen LLM planner through *in-context adaptation*, enabling online plan refinement without parameter updates. The planning process follows a structured *filter–then–evaluate* decision logic, as summarized in Algorithm 1. At each decision step, the LLM proposes a candidate action, which is then sequentially processed by the two critics.

First, the Feasibility Critic (C_F) determines whether the proposed action satisfies hard physical and logical constraints. Infeasible actions are im-

Replanning Example

System Prompt for Replanning:
You are a task planning assistant. Your goal is to complete: `put pancake in microwave`. The current sub-goal is: `find pancake`.

Recent Failed Attempts:
I tried `[walk] <sofa> (1)` but it received a low score of 3. It was a bad move.
I tried `[walk] <desk> (2)` but it received a low score of 4. It was also a bad move.

Based on this, provide a better, single next action to take. Do not repeat the failed actions. Your action must follow the format `[action] <object> (id)`.

Figure 3: An illustrative prompt template for in-context adaptation. When an action receives a low score from C_Q, the history of failures is explicitly added to the LLM's context to guide it towards a better alternative.

mediately rejected and never executed. For actions deemed feasible, the Quality Critic (C_Q) evaluates their strategic merit within the safe action set. Actions whose quality score falls below a threshold τ are rejected as suboptimal.

When an action is rejected by either critic, DDCG performs *in-context adaptation* by updating the LLM's prompt with explicit diagnostic feedback indicating the reason for rejection (Figure 3). This feedback is accumulated in the action history and conditions subsequent proposals, enabling the LLM to refine its behavior over time without retraining.

Only actions that pass both feasibility and quality checks are executed in the environment. Their outcomes and associated quality scores are appended to the planning context, closing the online feedback loop.

4.4 Theoretical Analysis

We provide a theoretical grounding of DDCG by formulating it as a critic-regularized optimization problem. Starting from a base LLM planner policy π^{llm} defined over the full action space $\mathcal{A}$, the Feasibility Critic C_F induces a state-dependent safe action set $\mathcal{A}_{\text{safe}}(s)$ by enforcing hard physical and logical constraints, while the Quality Critic C_Q provides a dense reward signal $r_Q(s,a)$ over feasible actions.

Formally, for a given state s, the objective of DDCG is to derive an improved policy π^{ddcg} that maximizes the expected quality reward while remaining close to the base policy:

$$\pi^{\text{ddcg}} = \arg\max_{\pi} \mathbb{E}_{a\sim\pi(\cdot|s)}[r_Q(s,a)] - \beta\, D_{\text{KL}}\Big(\pi(\cdot|s) \,\|\, \pi^{\text{llm}}(\cdot|s)\Big), \tag{1}$$

subject to the hard constraint that the policy assigns zero probability to actions outside $\mathcal{A}_{\text{safe}}(s)$.

Under this constrained optimization problem, the optimal policy admits a closed-form solution.

Lemma 1. *The optimal policy that solves the above optimization problem is given by*

$$\pi^*(a|s) = \begin{cases} \dfrac{1}{Z(s)}\,\pi^{llm}(a|s)\exp\left(\dfrac{r_Q(s,a)}{\beta}\right), & a \in \mathcal{A}_{safe}(s), \\ 0, & a \notin \mathcal{A}_{safe}(s), \end{cases} \tag{2}$$

where

$$Z(s) = \sum_{a'\in\mathcal{A}_{safe}(s)} \pi^{llm}(a'|s)\exp\left(\frac{r_Q(s,a')}{\beta}\right)$$

is the normalization factor.

This form makes explicit how feasibility and quality influence the policy through distinct mechanisms: feasibility restricts the support of the policy, while quality reweights actions within the safe set.

Corollary 2. *The resulting policy π^* is a policy improvement over the base policy π^{llm} with respect to the value function induced by r_Q, i.e.,*

$$Q^{\pi^*}(s,a) \geq Q^{\pi^{llm}}(s,a) \quad \textit{for all } (s,a) \in \mathcal{S}\times\mathcal{A}_{safe}(s).$$

The proof follows directly from standard results in critic-regularized reinforcement learning and is provided in Appendix A.

5 Experiments

We evaluate DDCG through a comprehensive set of experiments designed to assess both its empirical performance and its ability to resolve Signal Confounding in LLM-based embodied planning. We first introduce the experimental setup and evaluation protocol (Section 5.1), followed by the main results on VirtualHome and ScienceWorld (Section 5.2). We then present ablation studies that validate the necessity and non-redundancy of the decoupled critics (Section 5.3). Finally, we provide decision-level analyses and qualitative case studies to illustrate how decoupled feedback leads to more reliable planning behavior (Sections 5.4 and 5.5).

5.1 Experimental Settings

Environments. We evaluate DDCG on two challenging embodied agent benchmarks with complementary task characteristics: **VirtualHome** (Puig et al., 2018), a 3D household simulation emphasizing long-horizon planning under complex object interactions, and **ScienceWorld** (Wang et al., 2022), a text-based environment focusing on procedurally structured scientific tasks. For VirtualHome, following prior work (Li et al., 2022), we adopt the

Method	In-Distribution		Novel Scenes		Novel Tasks	
	Exec.	SR	Exec.	SR	Exec.	SR
MLDT (Wu et al., 2024)	38.0	34.0	32.0	31.0	34.0	33.0
LID-ADG (Li et al., 2022)	–	46.7	–	32.2	–	25.5
ProgPrompt (Singh et al., 2023)	87.3	82.3	38.7	32.3	49.7	49.0
Tree Planner (Hu et al., 2024)	–	–	**89.3**	41.7	**90.3**	52.3
Inner Monologue (Huang et al., 2022b)	79.7	79.3	54.3	53.3	47.3	46.0
DGAP-GPT4 (Qian et al., 2025)	93.3	88.0	71.7	62.7	73.7	72.2
DDCG-GPT4 (Ours)	**95.0**	**94.1**	**75.6**	**73.6**	**83.3**	**81.8**

Table 1: Main results on the VirtualHome benchmark. DDCG significantly outperforms all baselines, demonstrating the effectiveness of decoupled criticism.All values are reported in percentage.

Model	Short	Medium	Long	Overall
TDT (Chen et al., 2021)	28.4	10.4	6.1	14.7
SayCan (Ahn et al., 2022)	43.8	36.6	23.7	33.8
ReAct (Yao et al., 2023)	48.8	44.0	21.1	36.4
Reflexion (Shinn et al., 2023)	71.5	35.4	30.2	45.3
SwiftSage (Lin et al., 2023)	**92.2**	77.8	83.0	84.7
DGAP (Qian et al., 2025)	90.8	81.8	84.5	85.9
DDCG (Ours)	91.2	**84.3**	**85.7**	**87.1**

Table 2: Average scores of different methods on ScienceWorld.

standard *In-Distribution*, *Novel Tasks*, and *Novel Scenes* evaluation settings to assess generalization under increasing distribution shift.

Evaluation Metrics. To appropriately assess performance in each benchmark, we employ metrics tailored to their distinct characteristics. For VirtualHome, we use two standard metrics (Qian et al., 2025). Success Rate (SR) is a task-level metric that evaluates the final outcome of the entire plan. In contrast, Executability (Exec) is a step-level metric that measures the fraction of actions in a generated plan that are valid. For ScienceWorld, we use the average final score as the evaluation metric.

Training Settings. For our method, DDCG, we use **GPT-4** (OpenAI, 2023) as the backbone LLM planner. The critic models, C_F and C_Q, are implemented as lightweight, fine-tuned RoBERTa-base models (Liu et al., 2020). The data generation process utilizes a fine-tuned FLAN-T5-Large model (Chung et al., 2022). Implement details are provided in Appendix B

Baselines. We compare DDCG against strong baselines spanning multiple guidance paradigms to evaluate its effectiveness in resolving Signal Confounding. Our primary comparison is with the state-of-the-art **step-level guidance** method *DGAP* (Qian et al., 2025), which relies on a single, confounded score. We further include representative baselines from other paradigms, including the **search-based** *Tree Planner* (Hu et al., 2024), the **self-feedback** method *Inner Monologue* (Huang et al., 2022b), and the **demonstration-driven** agent *SwiftSage* (Lin et al., 2023). Additional planning approaches such as *ProgPrompt*, *MLDT*, and *LID-ADG* are also included for completeness. All methods use the same backbone LLM where applicable and are evaluated under identical conditions.

5.2 Main Results

As shown in Table 1, our method not only achieves a leading performance in success rate (SR) but also demonstrates a clear advantage in executability (Exec). Compared to all baselines, DDCG exhibits a notably smaller gap between SR and Exec, indicating that valid actions selected by the planner are more likely to contribute to task completion. This consistency directly reflects the effectiveness of our decoupled feedback mechanism in resolving the Signal Confounding problem.

On **In-Distribution** tasks, DDCG achieves a 95.0% success rate, significantly outperforming the strongest baseline, DGAP (88.0%). This advantage becomes more pronounced under generalization. On **Novel Scenes** and **Novel Tasks**, DDCG maintains high success rates of 73.6% and 81.8%,

Setting	Model	Exec. (%)	SR (%)
In-Distribution	DDCG (Ours)	**95.0**	**94.1**
	w/o C_F	89.4	82.3
	w/o C_Q	93.1	90.0
Novel Scenes	DDCG (Ours)	**75.6**	**73.6**
	w/o C_F	74.2	70.1
	w/o C_Q	70.5	63.2
Novel Tasks	DDCG (Ours)	**83.3**	**81.8**
	w/o C_F	78.9	75.0
	w/o C_Q	82.5	80.0

Table 3: Ablation study results on the VirtualHome benchmark.

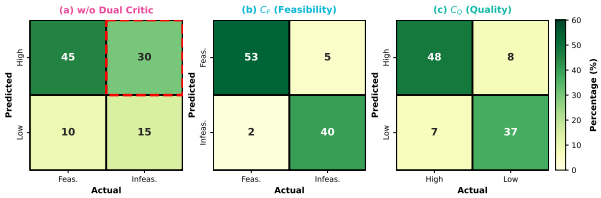

Figure 4: Confusion matrices illustrating decision-level error profiles under different ablation settings.

respectively, despite increased environmental uncertainty. In contrast, methods that emphasize feasibility without strategic evaluation struggle to generalize. For example, Tree Planner (Hu et al., 2024) achieves high executability (90.3%) on novel tasks but exhibits a much lower success rate (52.3%). This divergence between high Exec and low SR is a direct manifestation of Signal Confounding, illustrating that ensuring feasibility alone is insufficient for robust performance in unfamiliar settings.

DDCG also demonstrates strong performance improvements on the ScienceWorld benchmark. As shown in Table 2, it attains the highest overall score of 87.1, surpassing strong baselines such as DGAP (85.9) and SwiftSage (84.7). While SwiftSage performs competitively on short-horizon tasks, DDCG establishes a clear advantage on **Medium** and **Long** tasks. This trend highlights the benefit of decoupled feedback as task horizons grow longer: by distinguishing infeasible actions from low-quality but valid ones, DDCG prevents early errors from propagating and accumulating, resulting in more coherent and successful long-horizon plans.

5.3 Ablation Study

To verify that decoupling feasibility and strategic quality is both necessary and non-redundant for resolving Signal Confounding, we conduct ablation studies on the VirtualHome benchmark. We compare the full DDCG framework against two

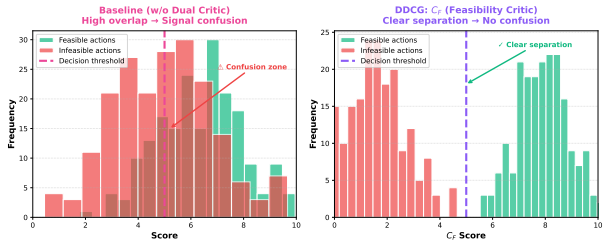

Figure 5: Score distributions for feasible versus infeasible actions.

variants: **DDCG w/o** C_F, which removes feasibility supervision and retains only quality guidance, and **DDCG w/o** C_Q, which applies feasibility filtering without strategic evaluation.

As shown in Table 3, removing either critic leads to a substantial performance degradation, but in qualitatively different ways. Removing the Feasibility Critic (C_F) primarily harms plan validity, causing a notable drop in Executability (e.g., from 95.0% to 89.4% In-Distribution) and a more severe decline in Success Rate (from 94.1% to 82.3%). This confirms that C_F is essential for enforcing hard physical and logical constraints.

In contrast, removing the Quality Critic (C_Q) leaves Executability largely intact but significantly degrades Success Rate, especially in the challenging *Novel Scenes* setting (from 73.6% to 63.2%). This indicates that feasibility alone is insufficient for effective planning: without strategic guidance, the agent frequently executes actions that are valid but inefficient or incoherent over long horizons.

Beyond aggregate performance drops, we further analyze how removing each critic alters the error profile at the decision level. As shown in Figure 4, the baseline exhibits a high false-positive rate, frequently misclassifying infeasible actions as acceptable. In contrast, the Feasibility Critic substantially reduces such errors, while the full DDCG framework achieves a clear separation between feasible and infeasible actions.

Taken together, these ablations demonstrate that C_F and C_Q address complementary and non-overlapping failure modes. Their decoupled integration is therefore critical for precise error attribution and for effectively resolving Signal Confounding.

5.4 Score-space Analysis

To analyze how supervision signals interact with the decoupled critics, we examine the score distributions assigned to feasible and infeasible actions.

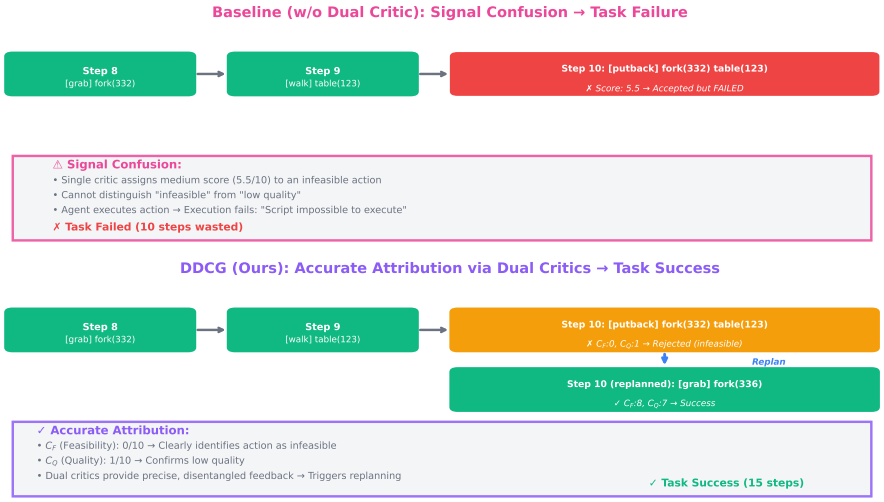

Figure 6: Qualitative comparison of a critical planning step. The baseline misattributes an infeasible action as medium quality due to Signal Confounding, leading to execution failure. In contrast, DDCG uses its decoupled critics to accurately identify the infeasibility, reject the action pre-execution, and trigger a successful replan.

As shown in Figure 5, baseline methods that rely on a single scalar score exhibit substantial overlap between the two classes, forming a broad confusion region where infeasible actions receive moderate scores. This overlap directly reflects the signal confounding problem, as feasibility violations and low-quality but valid actions become indistinguishable under a single scoring function.

In contrast, DDCG induces a structured separation of signals. The Feasibility Critic assigns consistently low scores to infeasible actions and high scores to feasible ones, resulting in a clear margin between the two distributions. This separation enables the planner to reliably reject infeasible actions prior to execution, eliminating a major source of failure observed in single-score baselines.

5.5 Qualitative Analysis

We analyze representative planning trajectories to illustrate how DDCG resolves Signal Confounding at decision time. Figure 6 compares DDCG with a baseline that relies on a single, scalar feedback signal.

In the illustrated VirtualHome task, the agent proposes the action `[putback] <cutleryfork> <kitchentable>`, which is semantically plausible but physically infeasible due to unmet environmental constraints. The baseline assigns a medium score (5.5/10) to this action, conflating infeasibility with moderate quality. As a result, the action is accepted and executed, leading to an immediate execution failure and termination of the episode.

Under the same conditions, DDCG produces disentangled feedback. The Feasibility Critic (C_F) correctly classifies the action as infeasible ($C_F = 0$), while the Quality Critic (C_Q) assigns a low quality score, reflecting its poor strategic value. This explicit attribution causes the action to be rejected *prior* to execution and triggers replanning, after which the agent selects a feasible alternative and completes the task successfully.

6 Conclusion

This work identifies **Signal Confounding** as a fundamental structural limitation in LLM-based embodied planning, arising from collapsing multi-dimensional state–action evaluations into a single scalar feedback signal. To address this critical issue, we propose **Decoupled Dual-Critic Guidance (DDCG)**, which explicitly separates feasibility and strategic quality into two complementary critics that provide structured decision-time feedback. By enforcing hard execution constraints through the Feasibility Critic and shaping preferences among feasible actions through the Quality Critic, DDCG restores reliable error attribution and effectively mitigates various failure modes caused by confounded signals. Experimental results across two embodied planning benchmarks demonstrate that this decoupling approach leads to more consistent executability, improved task success rates, and greater robustness under generalization conditions, suggesting that structured feedback is a key ingredient for achieving robust and reliable LLM-based planning.

Limitations

While DDCG improves the reliability of LLM-based embodied planning by resolving signal confounding, it has several limitations. The framework relies on the accuracy of the learned critics: errors in the Feasibility Critic may lead to overly conservative behavior, while miscalibration of the Quality Critic can still result in suboptimal yet feasible decisions. In addition, DDCG assumes that the planner's textual context provides a sufficiently informative state representation; we do not explicitly model partial observability or belief uncertainty, which may limit performance in environments with severe perception ambiguity. Finally, our evaluation is conducted on two simulated benchmarks, and further studies are required to assess scalability to more complex environments, longer horizons, or real-world robotic systems.

Ethical considerations

We take ethical considerations seriously and conduct this research in accordance with established academic and ethical standards. This paper studies planning and feedback mechanisms for LLM-based embodied agents in simulated environments, with the goal of improving reliability and error attribution during task execution. Our proposed framework does not introduce prompts, training procedures, or objectives that encourage the generation of harmful, deceptive, or unsafe content; instead, it focuses on evaluating action feasibility and strategic quality to prevent invalid or unsafe behaviors during planning. All experiments are performed in simulation using publicly available benchmarks and models that are widely adopted by the research community. No human participants are involved as evaluators or subjects in this study, and we report our methods and findings accurately and transparently.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2022. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

Eitan Altman. 2021. *Constrained Markov decision processes*. Routledge.

Chenjia Bai, Huazhe Xu, and Xuelong Li. 2024. Embodied-ai with large models: research and challenges. *SCIENTIA SINICA Informationis*, 54(9).

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: reinforcement learning via sequence modeling. NIPS '21.

Zhenfang Chen, Delin Chen, Rui Sun, Wenjun Liu, and Chuang Gan. 2025. Autonomous agents from automatic reward modeling and planning. In *The Thirteenth International Conference on Learning Representations*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Xian Fu, Min Zhang, Jianye Hao, Peilong Han, Hao Zhang, Lei Shi, and Hongyao Tang. 2024. What can vlms do for zero-shot embodied task planning? *Workshop on Large Language Models and Cognition at the 41st International Conference on Machine Learning*.

Shivank Garg, Ayush Singh, Shweta Singh, and Paras Chopra. 2025. Ipo: Your language model is secretly a preference classifier. *arXiv preprint arXiv:2502.16182*.

Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. 2024. Tree-planner: Efficient close-loop task planning with large language models. *Preprint*, arXiv:2310.08582.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. 2022b. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, and Yuke Zhu. 2022. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

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. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. *Preprint*, arXiv:1806.07011.

Haofu Qian, Chenjia Bai, Jiatao Zhang, Fei Wu, Wei Song, and Xuelong Li. 2025. Discriminator-guided embodied planning for LLM agent. In *The Thirteenth International Conference on Learning Representations*.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve. *Preprint*, arXiv:2407.18219.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *Preprint*, arXiv:2206.05802.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11499–11505.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *Preprint*, arXiv:2203.07540.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. 2024. Long-form factuality in large language models.

Yike Wu, Jiatao Zhang, Nan Hu, Lanling Tang, Guilin Qi, Jun Shao, Jie Ren, and Wei Song. 2024. MLDT: multi-level decomposition for complex long-horizon robotic task planning with open-source large language model. In *DASFAA (5)*, volume 14854 of *Lecture Notes in Computer Science*, pages 251–267. Springer.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2025. Agentrm: Enhancing agent generalization with reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19277–19290, Vienna, Austria. Association for Computational Linguistics.

Ruihan Yang, Fanghua Ye, Jian Li, Zhaopeng Tu, Xiaolong Li, Siyu Yuan, Yikai Zhang, and Deqing Yang. 2025. The lighthouse of language: Enhancing llm agents via critique-guided improvement. *arXiv preprint arXiv:2503.16024*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

A Theoretical Proofs

A.1 Proof of Lemma 1

Lemma 1. *The optimal policy that solves the constrained optimization problem is given by:*

$$\pi^*(a|s) = \begin{cases} \frac{1}{Z(s)}\,\pi^{\text{llm}}(a|s)\exp\left(\frac{r_Q(s,a)}{\beta}\right), & a \in \mathcal{A}_{\text{safe}}(s), \\ 0, & a \notin \mathcal{A}_{\text{safe}}(s), \end{cases} \tag{3}$$

where

$$Z(s) = \sum_{a' \in \mathcal{A}_{\text{safe}}(s)} \pi^{\text{llm}}(a'|s)\exp\left(\frac{r_Q(s,a')}{\beta}\right) \tag{4}$$

is the normalization factor.

Proof. For a given state s, the optimization problem is to find a policy $\pi(\cdot|s)$ that maximizes

$$\begin{aligned}\mathcal{L}(\pi) = &\sum_{a \in \mathcal{A}_{\text{safe}}(s)} \pi(a|s)\, r_Q(s,a) \\ &- \beta \sum_{a \in \mathcal{A}_{\text{safe}}(s)} \pi(a|s) \log \frac{\pi(a|s)}{\pi^{\text{llm}}(a|s)},\end{aligned} \tag{5}$$

subject to the normalization constraint

$$\sum_{a \in \mathcal{A}_{\text{safe}}(s)} \pi(a|s) = 1. \tag{6}$$

Introducing a Lagrange multiplier λ, the Lagrangian is

$$\mathcal{J}(\pi,\lambda) = \mathcal{L}(\pi) - \lambda\left(\sum_{a \in \mathcal{A}_{\text{safe}}(s)} \pi(a|s) - 1\right). \tag{7}$$

Taking the derivative with respect to $\pi(a|s)$ and setting it to zero yields

$$r_Q(s,a) - \beta\left(\log\frac{\pi(a|s)}{\pi^{\text{llm}}(a|s)} + 1\right) - \lambda = 0. \tag{8}$$

Solving for $\pi(a|s)$ gives

$$\log \pi(a|s) = \log \pi^{\text{llm}}(a|s) + \frac{r_Q(s,a)}{\beta} - 1 - \frac{\lambda}{\beta}, \tag{9}$$

and thus

$$\pi^*(a|s) = \pi^{\text{llm}}(a|s)\exp\left(\frac{r_Q(s,a)}{\beta}\right)\exp\left(-1 - \frac{\lambda}{\beta}\right). \tag{10}$$

Let $Z'(s) = \exp(1 + \lambda/\beta)$. Enforcing normalization,

$$\begin{aligned}1 &= \sum_{a' \in \mathcal{A}_{\text{safe}}(s)} \pi^*(a'|s) \\ &= \frac{1}{Z'(s)} \sum_{a' \in \mathcal{A}_{\text{safe}}(s)} \pi^{\text{llm}}(a'|s)\exp\left(\frac{r_Q(s,a')}{\beta}\right),\end{aligned} \tag{11}$$

which implies $Z'(s) = Z(s)$ and completes the proof. □

A.2 Proof of Corollary 2

Corollary 2. *The policy π^* is a policy improvement over π^{llm} with respect to the value function induced by r_Q.*

Proof. Let $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)]$. We aim to show that $V^{\pi^*}(s) \geq V^{\pi^{\text{llm}}}(s)$ for all s.

By the policy improvement theorem, it suffices to verify that

$$\sum_a \pi^*(a|s) Q^{\pi^{\text{llm}}}(s,a) \geq \sum_a \pi^{\text{llm}}(a|s) Q^{\pi^{\text{llm}}}(s,a). \tag{12}$$

Since π^* up-weights actions with larger $r_Q(s,a)$ relative to π^{llm}, it increases the expected value under $Q^{\pi^{\text{llm}}}$. Expanding the value difference yields

$$\begin{aligned}&V^{\pi^*}(s) - V^{\pi^{\text{llm}}}(s) \\ &\quad = \mathbb{E}_{a\sim\pi^*(\cdot|s)}\left[Q^{\pi^{\text{llm}}}(s,a) - V^{\pi^{\text{llm}}}(s)\right] \\ &\quad + \mathbb{E}_{s'\sim P}\left[\gamma\left(V^{\pi^*}(s') - V^{\pi^{\text{llm}}}(s')\right)\right],\end{aligned} \tag{13}$$

which is non-negative by recursion, completing the proof. □

Bridging Theory and Practice. This derivation establishes the formal foundation for DDCG, framing embodied planning as a constrained policy optimization problem. While Lemma A.1 characterizes the optimal regularized policy π^*, our prompt-level replanning procedure (Algorithm 1) serves as a practical and efficient realization of this framework. Specifically, the Feasibility Critic (C_F) operationalizes the hard constraints by defining the safe action set $\mathcal{A}_{\text{safe}}$, while the Quality Critic (C_Q) provides the "strategic gradient" that guides the LLM's in-context adaptation toward the optimal distribution.

B Implementation Details

This section provides the implementation details necessary to reproduce our experiments, including the data generation model used for synthesizing supervision signals and the training procedure for the dual critics in DDCG.

B.1 Data Generation Model

Figure 7 provides an overview of the data generation and label decoupling pipeline used to construct supervision for the dual critics. To generate suboptimal action candidates for training the Quality Critic, we finetune a FLAN-T5-Large model on expert trajectories in a sequence-to-sequence manner. Our data generation procedure follows the

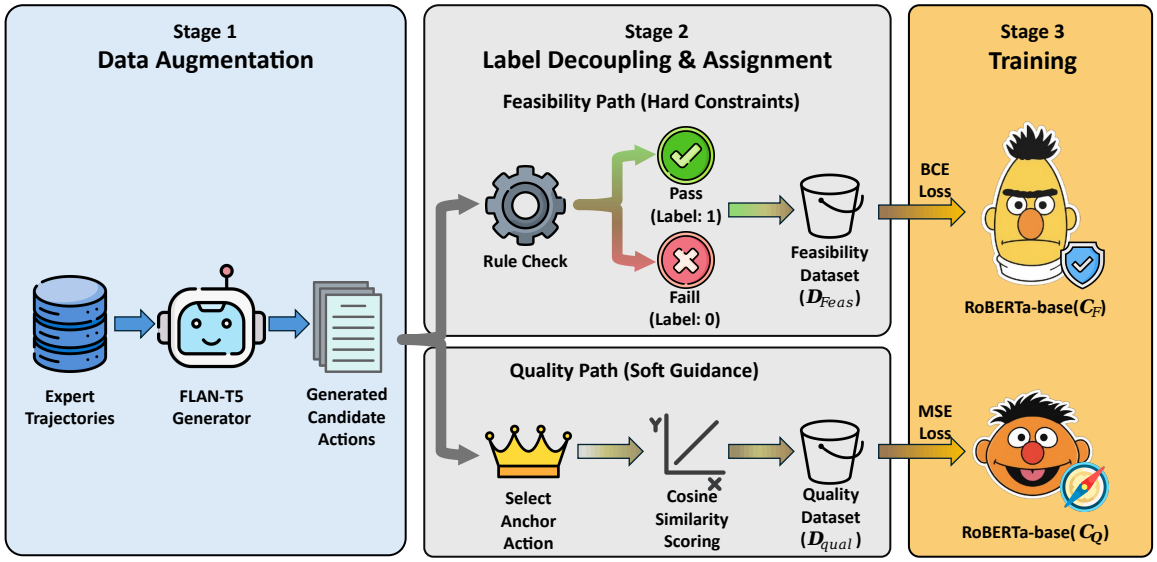

Figure 7: Overview of the data generation and label decoupling process used for training the Feasibility Critic (C_F) and the Quality Critic (C_Q).

step-level action generation strategy introduced in DGAP (Qian et al., 2025), which uses a fine-tuned language model to synthesize diverse candidate actions beyond expert demonstrations.

Concretely, the input to the data generation model consists of the task description and the action history, and the target is the next expert action. We adopt the standard FLAN-T5-Large architecture and pretrained weights provided by the original implementation, and finetune the model to capture domain-specific action patterns from expert trajectories. During data synthesis, the fine-tuned model generates multiple candidate actions for a given state via beam search, producing a set of plausible but potentially suboptimal actions that enrich the supervision signal for training the Quality Critic.

Semantic Similarity Scoring. To assign quality scores to the generated suboptimal actions, we rank candidate actions based on their semantic similarity to a high-quality reference action. For each state, the highest-ranked action produced by beam search is treated as the reference action $a^\star$. Each remaining candidate action a is embedded using a pretrained sentence encoder, and similarity is computed via cosine similarity:

$$\mathrm{Sim}(a, a^\star) = \frac{\langle \mathbf{e}(a), \mathbf{e}(a^\star) \rangle}{\|\mathbf{e}(a)\|_2 \, \|\mathbf{e}(a^\star)\|_2}, \tag{14}$$

where $\mathbf{e}(\cdot)$ denotes the sentence embedding of an action description. Candidate actions are ranked according to $\mathrm{Sim}(a, a^\star)$ and discretized into quality score bins following this ranking. The sentence encoder is fixed and not finetuned on the target environments.

Hyperparameter	Value
Base Model	FLAN-T5-Large
Max Source Length	1024 tokens
Max Target Length	16 tokens
Optimizer	AdamW
Learning Rate	1e-4
Training Epochs	8
Per-Device Batch Size	8
Gradient Accumulation Steps	2
Effective Batch Size	32
Random Seed	42

Table 4: Hyperparameters for FLAN-T5 finetuning on expert data.

Finetuning of the data generation model is performed using the AdamW optimizer with a learning rate of 1e-4 for 8 epochs. Table 4 summarizes the key hyperparameters used for training. Example input–output formats for ScienceWorld and VirtualHome are shown in Listing 8 and Listing 9, respectively. While DGAP employs a single discriminator to assign step-level scores, our framework further decomposes supervision by separately modeling feasibility and strategic quality, enabling more precise error attribution during planning.

B.2 Critic Models Training

The Feasibility Critic (C_F) and the Quality Critic (C_Q) are implemented using RoBERTa-base with a linear prediction head. The two critics share the same backbone architecture but differ in their out-

Hyperparameter	Value
Backbone Model	RoBERTa-base
Loss (C_F)	Binary Cross-Entropy
Loss (C_Q)	Mean Squared Error
Max Sequence Length	256
Optimizer	AdamW
Learning Rate	1e-5
Weight Decay	0.01
Warmup Ratio	0.1
Batch Size	32
Training Epochs	5
Random Seed	42

Table 5: Hyperparameters for critic model finetuning.

```
ScienceWorld:
{"input": "Your task is to boil water. For compounds without a boiling point, combusting the substance is also
acceptable.", "Action": "open door to kitchen"}
{"input": "Your task is to boil water. For compounds without a boiling point, combusting the substance is also acceptable.
10. open door to kitchen.", "Action": "go to kitchen"}
{"input": "Your task is to boil water. For compounds without a boiling point, combusting the substance is also acceptable.
10. open door to kitchen. 9. go to kitchen.", "Action": "pick up thermometer"}
{"input": "Your task is to boil water. For compounds without a boiling point, combusting the substance is also acceptable.
10. open door to kitchen. 9. go to kitchen. 8. pick up thermometer.", "Action": "open cupboard"}
```

Figure 8: FLAN-T5 finetuning data examples for ScienceWorld.

```
VirtualHome:
{"input": "Task: Wash dishes", "action": "[Walk] <kitchen> (1)"}
{"input": "Task: Wash dishes. [Walk] <kitchen> (1)", "action": "[Walk] <sink> (1)"}
{"input": "Task: Wash dishes. [Walk] <kitchen> (1), [Walk] <sink> (1)", "action": "[Find] <faucet> (1)"}
```

Figure 9: FLAN-T5 finetuning data examples for VirtualHome.

put heads and training objectives: C_F performs binary classification to predict action feasibility, while C_Q performs regression to estimate action quality.

Both critics are initialized from pretrained RoBERTa-base weights and optimized using the AdamW optimizer with a learning rate of `1e-5` and a weight decay of `0.01`. A linear learning rate scheduler with a warmup ratio of 0.1 is used. Training is conducted for 5 epochs with a batch size of 32. The Feasibility Critic is trained using Binary Cross-Entropy loss, and the Quality Critic is trained using Mean Squared Error loss. Key hyperparameters are summarized in Table 5.

B.3 Training Data Statistics

For critic training, we construct a tripartite dataset consisting of expert actions, hard negative samples, and suboptimal actions generated by the data generation model. In ScienceWorld, the dataset includes approximately 40k expert samples, 40k hard negative samples, and 400k suboptimal samples. In VirtualHome, the dataset consists of approximately 30k expert samples, 30k hard negative samples, and 350k suboptimal samples. The data formats for these samples are illustrated in Listing 10 and Listing 11.

C Prompt Templates

This section provides the detailed prompt templates used in our DDCG framework for both the VirtualHome (VH) and ScienceWorld (SW) environments.

C.1 VirtualHome Prompts

General Task Planning This template is a monolithic prompt that provides the LLM with all necessary context in a single block to directly generate a full sequence of actions. The structure includes sections for predefined rules, the initial state of the environment, a list of available actions, the specific task instruction, and finally a section for the LLM to generate its plan.

Reasoner + Planner This template employs a hierarchical, two-step approach. First, a 'Reasoner' component is prompted to decompose the high-level task into a sequence of more abstract subgoals. Subsequently, a 'Planner' component is prompted with a specific subgoal from the sequence to generate the corresponding low-level, executable actions required to complete that step.

Replanning This template is designed for error correction and uses a history of failed attempts as explicit negative feedback. The prompt contains a dedicated section for previously unsuccessful plans. The LLM is then prompted to generate a new plan with the explicit instruction to avoid repeating the previous mistakes, thereby guiding the model toward a valid solution.

C.2 ScienceWorld Prompts

General Task Planning This template is structured for state-aware, step-by-step interaction with the environment. It provides the LLM with a comprehensive history of its past actions, including the resulting observations and rewards. The prompt also includes the current environmental state, the agent's inventory, and a list of visited locations to maintain full context. Based on this rich historical and current state information, the LLM is prompted to generate the single best action to take for the next step.

Replanning This template is designed for structured error correction by providing the LLM with rich diagnostic information. The prompt includes a detailed history of both 'Failed Attempts and Diagnostic Feedback' as well as 'Recent Successful Actions' with their associated quality scores. It also provides a list of 'Available Action Types'

SW-Hard Negative Samples:
{"input": " Your task is to use chemistry to create violet paint. When you are done, focus on the violet paint. 10. open door to greenhouse. 9. go to greenhouse. 8. open door to hallway. 7. go to hallway.Action: pour cup containing red paint in desk in bowl", "Score": "10"}
SW-Expert Data:
{"input": " Your task is to determine which of the two inclined planes (unknown material D, unknown material H) has the most friction...Action: sleep 112", "Score": "0"}
SW-Hard Negative Samples:
{"input": " Your task is to determine which of the two inclined planes (titanium, sandpaper) has the least friction. Action: deactivate stopwatch in inventory", "Score": "3"}

Figure 10: Critic training data examples for ScienceWorld.

VH-Hard Negative Samples:
{"input": "Task: Sleep. Lie down in a bed at night and fall asleep. Action: [Grab] <sofa> (769)" "Score": "0"}
VH-Expert Data:
{"input": "Task: Listen to music. [Walk] <living_room> (1). Action: [Walk] <radio> (1)", "Score": "10"}
VH-Suboptimal Data:
{"input": "Your task is to take bath. Action: [Walk] <bathroom> (1)", "Score": "4"}

Figure 11: Critic training data examples for VirtualHome.

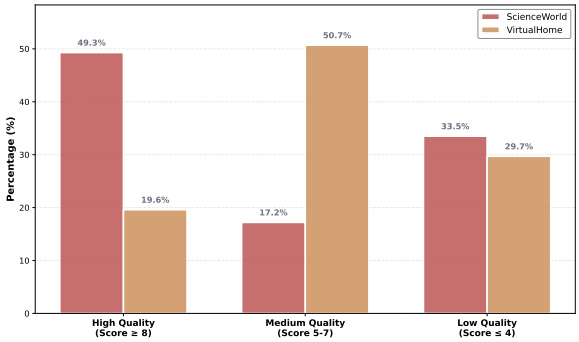

Figure 12: Quality category distribution of the generated data for ScienceWorld and VirtualHome.

and 'Important Rules' to constrain the generation space. The LLM is then given an explicit, multi-point instruction to generate a "BETTER" action that satisfies several constraints, such as avoiding past mistakes, ensuring physical executability, and achieving a minimum quality score.

This section provides additional experimental analyses to complement the main results presented in Section 5.2. We report detailed performance breakdowns on the ScienceWorld benchmark and further analyze the properties of the generated supervision data used to train the critics.

C.3 Detailed Results on ScienceWorld

Table 6 reports the average scores of different methods on the ScienceWorld benchmark. Overall, DDCG achieves the highest average score among the compared methods. To better understand this performance, we analyze results stratified by task length.

On **Short** tasks, several methods perform competitively. In particular, SwiftSage slightly outperforms DDCG, indicating that for short-horizon problems with limited error accumulation, the benefit of sophisticated feedback mechanisms is less pronounced. However, as task length increases, the advantages of decoupled guidance become increasingly evident.

On **Medium** tasks, DDCG establishes a clear performance margin over both DGAP and SwiftSage. This trend is further amplified on **Long** tasks, where DDCG maintains the highest scores among all baselines. These results suggest that long-horizon procedural reasoning is especially vulnerable to cumulative planning errors, and that resolving Signal Confounding becomes critical as task complexity increases.

C.4 Quality Distribution of Supervision Data

To further contextualize the training of the Quality Critic, we analyze the distribution of quality labels in the generated supervision data. Figure 12 shows the proportion of high-, medium-, and low-quality action samples for ScienceWorld and VirtualHome.

The two environments exhibit markedly different distributions. In ScienceWorld, the data are strongly polarized, with a large fraction of samples assigned either high or low quality scores. This reflects the strictly procedural nature of ScienceWorld tasks, where incorrect steps or invalid actions often lead to immediate failure. In contrast, VirtualHome displays a more balanced distribution with a substantial proportion of medium-quality samples, consistent with the presence of multiple feasible but variably efficient action choices in household environments.

D Additional Experimental Results

These complementary distributions provide suitable supervision signals for training the critics. Polarized quality labels in ScienceWorld facilitate learning clear decision boundaries, while the smoother distribution in VirtualHome encourages fine-grained quality discrimination among feasible actions. Together, this analysis highlights how the data generation process aligns with the structural differences between the two environments.

Task Type	*Len	TDT	SayCan	ReAct	Reflexion	SwiftSage	DGAP	DDCG
1-1 (L)	107.7	0.71	33.06	3.52	4.22	97.04	100.0	**100.0**
1-2 (L)	78.6	0.44	10.39	13.70	10.61	87.04	**92.8**	87.1
1-3 (L)	88.9	3.88	3.88	7.78	7.78	72.78	74.0	**83.0**
1-4 (L)	75.2	0.55	0.37	9.88	0.92	100.0	**100.0**	94.2
2-1 (M)	21.4	6.16	26.37	7.19	5.92	99.17	100.0	**100.0**
2-2 (M)	35.2	6.43	8.03	6.10	28.59	**88.17**	80.17	87.2
2-3 (L)	65.0	19.87	17.41	22.37	22.37	**95.73**	88.33	94.2
3-1 (S)	13.6	40.55	52.14	56.0	**100.0**	88.67	91.5	83.0
3-2 (M)	20.8	14.26	22.50	54.33	17.45	55.33	58.0	**58.0**
3-3 (M)	25.6	10.16	99.56	76.19	72.54	71.90	78.57	**82.7**
3-4 (M)	29.0	21.65	47.76	88.81	70.22	77.86	**88.14**	87.71
4-1 (S)	14.6	41.93	22.87	26.67	64.93	100.0	100.0	**100.0**
4-2 (S)	8.8	55.76	58.18	80.0	87.27	100.0	100.0	**100.0**
4-3 (S)	12.6	27.82	20.87	53.33	16.42	91.67	**100.0**	90.2
4-4 (S)	14.6	47.15	31.43	27.50	100.0	100.0	100.0	**100.0**
5-1 (L)	69.5	6.89	9.92	9.06	7.33	**74.59**	73.14	72.57
5-2 (L)	79.6	11.86	13.93	18.57	13.0	93.93	90.57	**94.0**
6-1 (M)	33.6	15.10	47.81	51.04	**70.35**	49.40	57.4	59.0
6-2 (S)	15.1	15.70	39.26	58.89	70.67	100.0	100.0	**100.0**
6-3 (M)	23.0	5.25	19.72	40.74	15.77	91.48	92.43	**100.0**
7-1 (S)	7.0	30.0	80.0	60.0	100.0	95.0	100.0	**100.0**
7-2 (S)	7.0	8.43	67.50	67.50	84.37	85.0	85.71	**95.14**
7-3 (S)	8.0	8.34	50.0	50.0	83.0	93.33	92.71	**95.7**
8-1 (M)	40.0	3.86	20.91	27.67	2.58	89.0	100.0	**100.0**
8-2 (S)	16.3	8.0	16.0	8.0	8.0	**68.50**	38.50	47.5
9-1 (L)	97.0	2.53	21.94	40.50	50.63	75.0	75.0	**75.0**
9-2 (L)	84.9	14.66	32.26	44.0	**100.0**	70.0	83.33	78.6
9-3 (L)	123.1	9.12	13.67	41.0	70.62	60.0	71.43	**82.7**
10-1 (L)	130.1	1.51	67.53	25.70	50.90	**92.3**	87.71	88.5
10-2 (L)	132.1	1.29	59.45	16.80	23.69	77.60	78.0	**79.0**
Short	*11.76*	28.37	43.83	48.79	71.47	**92.22**	90.84	91.15
Medium	*28.58*	10.36	36.58	44.01	35.43	77.79	81.84	**84.32**
Long	*94.30*	6.11	23.65	21.07	30.17	83.0	84.52	**85.74**
Overall	*49.26*	14.66	33.82	36.43	45.34	84.68	85.91	**87.07**

Table 6: **Detail results on the ScienceWorld benchmark.** **Len* is the average length of the oracle agent's trajectories. In addition to overall results, we also report performance on three groups of **Len* (short, medium, long). The last six methods use GPT-4 as the base LLM for prompting.

General Task Planning prompt template for VH

 # Long-term Memory
Action Primitives:
from actions import walk <obj>, grab <obj>, switchon <obj>, switchoff <obj>, open <obj>, close <obj>, turnto <obj>, drink <obj>, putin <obj> <obj>,
putback <obj> <obj>

 # Rules:
remember if the key object INSIDE kitchencabinet, you should open the kitchencabinet first or the key object INSIDE room, you should walk to the room and different id represent different items, so note the id number.
remember you should grab only one item at a time and you can not open a cabinet that has been opened

 # Overall Task Goal:
{total_task_goal}

 # Completed Sub-goals:
{completed_sub_goals}

 # Recently Executed Actions (Optional, with scores):
{recent_actions_with_scores}

 # Short-term Memory
 # Few-shot Examples:
{few_shot_examples}

 # Current Environment State:
remember the key object locations and states: {current_env_state}

 # Current Task
The task goal: {current_sub_goal}

Figure 13: VH: General Task Planning Prompt.

Reasoner + Planner prompt template for VH

 # Now you are a task planning assistant, responsible for inferring the execution steps of a task.

 # You should mimic the provided examples and, based on the task objectives, understand the total task goal first, generate the next sub-task.

 # There are some examples:
{reasoner_few_shot_examples}
Imitate these examples to generate a step-by-step plan.

 # Task goal: {task_goal}

Now you are a task planning assistant. You should mimic the examples I provide and generate a sequence of actions based on the target instructions and environmental information. Pay attention to the task objectives and environmental information.

And remember if the key object INSIDE kitchencabinet, you should open the kitchencabinet first... (Rules from planner_rule)

 # There are some examples:
{planner_few_shot_examples}

Imitate these examples to generate an action list.

 # Now the task is: {reasoned_sub_task}
remember the key object locations and states: { current_env_state}

Figure 14: VH: Resoner and planner Prompt.

Replanning prompt template for VH

 # You are a task planning assistant. Your goal is to complete the task: '{ total_task_goal}'. The current sub-goal is: '{current_sub_goal}'.

 # Here are the only actions you are allowed to use:
walk <obj>, grab <obj>, switchon <obj>, switchoff <obj>, open <obj>, close <obj>, turnto <obj>, drink <obj>, putin <obj> <obj>, putback <obj> <obj>

 # Here are some important rules:
remember you should grab only one item at a time.
you can not open a cabinet that has been opened.
if the key object is INSIDE another object, you should open it first.

 # Here is the history of recent actions and why they were bad:
{failed_actions_history}

 # Based on this information, please provide a better, single next action to take FROM THE ALLOWED LIST.
 # Do not repeat the failed actions.
 # Just provide the single action command in the format like 'grab('kitchenknife(id:123)')'.

Figure 15: VH: Replanning Prompt.

General Task Planning prompt template for SW

 # Task Description:
{task_description}

 # Current Progress:
Time: {step_id}; Score: {current_score}/100;

 # Action History (with rewards):
Action history:
<extra_id_10> {action_9} (+{reward_9}) --> {observation_9}
<extra_id_9> {action_8} (+{reward_8}) --> {observation_8}
<extra_id_8> {action_7} (+{reward_7}) --> {observation_7}
...
<extra_id_1> {action_0} (+{reward_0}) --> {observation_0}

 # Current Environment:
{environment_observation}

 # Current Inventory:
{inventory}

 # Visited Locations:
{visited_rooms}

 # Next Action:
What action should you do next?

Figure 16: SW: General Task Planning Prompt.

Replanning prompt template for SW

 # You are a science experiment planning assistant

 # Task Goal:
{task_description}

 # Current Sub-goal:
{current_sub_goal}

 # Current Progress:
Score: {current_score}/100; Step: {step_id}

 # Environment Information:
Location: {current_location}
Environment: {environment_observation}
Inventory: {inventory}

 # Recent Successful Actions (with quality scores):
{recent_successful_actions_with_scores }

 # Failed Attempts and Diagnostic Feedback:
{failed_actions_history}

 # Available Action Types:
- teleport <location>: directly move to a room
- open <object>: open a container or door
- close <object>: close a container or door
- pick up <object>: take an object
- put <object> in <container>: place object in container
- activate <device>: turn on a device
- deactivate <device>: turn off a device
- pour <container> in <target>: pour contents
- mix <object> in <container>: mix substances
- focus on <object>: examine object for task completion
- wait: pause for time-dependent changes
- examine <object>: inspect an object

Figure 17: SW: Replanning Prompt (Part 1).

Replanning prompt template for SW

 # Important Rules:
- You can only manipulate one object at a time
- You must open containers before accessing their contents
- Broken devices cannot be activated
- Focus on task-relevant objects mentioned in the goal

 # Your Task Now:
Based on the failed attempts and diagnostic feedback above,
provide a BETTER action that:
1. Avoids previously failed strategies
2. Is physically executable in the current state
3. Makes meaningful progress toward the task goal
4. Achieves a quality score ≥ 4.0

Think step-by-step:
1. What went wrong with previous attempts?
2. What alternative approaches are available?
3. Which action would maximize progress?

Provide ONLY the single action command in this format:
Action: [your action]

Figure 18: SW: Replanning Prompt (Part 2).