DyRef: Dynamic Reflection Framework via Graph-Based Complexity for Robotic Planning

Anonymous Author(s)

Abstract— **Robotic planning tasks often involve diverse complexities, which make adaptive improvement through reflection particularly challenging. Existing LLM-based approaches typically rely on fixed routines, lacking the ability to adjust to task-specific complexity and often leading to redundant reflections. To address this, we propose DyRef, a dynamic reflection framework that models tasks as a Diagnostic Graph, measures tasks complexity through structural factors, and routes them through a Reflection Toolkit via a learned Routing Policy network. This design enables tailored reflection strategies that reduce redundancy and improve reasoning efficiency. Experiments in AlfWorld and on real-world robotic platforms show that DyRef improves success rates by 38.0% reducing redundant reflections by 64.4%. Project webpage (no author information) : https://DyRef.github.io/**

I. INTRODUCTION

Recent advancements in robotic task planning have increasingly leveraged large language models (LLMs) due to their ability to generate coherent multi-step plans [1], [2]. While LLM-based planning has proven effective in complex tasks, it still suffers from issues like hallucinations and logical inconsistencies [3]. To address these limitations, reflection mechanisms have emerged [4], [5], allowing robots to revisit and correct past decisions, transforming open-loop systems into more reliable closed-loop systems. However, existing reflection methods predominantly rely on pre-defined prompts or fixed routines, which restrict them to a narrow set of heuristics. This one-size-fits-all approach overlooks the diverse characteristics of tasks, causing reflection to be misaligned with the underlying task complexity and leading to either insufficient reflection or unnecessary overthinking [6].

Building on these observations, recent studies [7], [8] attempt to quantify task complexity and then decide the reflection level (e.g., simple vs. complex) based on this assessment (see Figure 1). This improves over fixed routines by allowing coarse adaptation to task complexity, but in real-world scenarios, task complexity is inherently multi-dimensional, involving aspects such as long-horizon dependencies, spatial reasoning, or intricate object interactions [9]. A single complexity metric fails to capture these diverse aspects, leading to inappropriate reflection, unnecessary overhead, and the accumulation of errors [10]. We therefore ask: **How can a robot dynamically adapt reflection with a multi-dimensional understanding of task complexity?**

To overcome such limitations, consider the medical domain. A doctor evaluates symptoms from multiple perspectives to build a comprehensive understanding of the illness. Based on this diagnosis, the doctor formulates a treatment strategy, which specifies how different tools—such

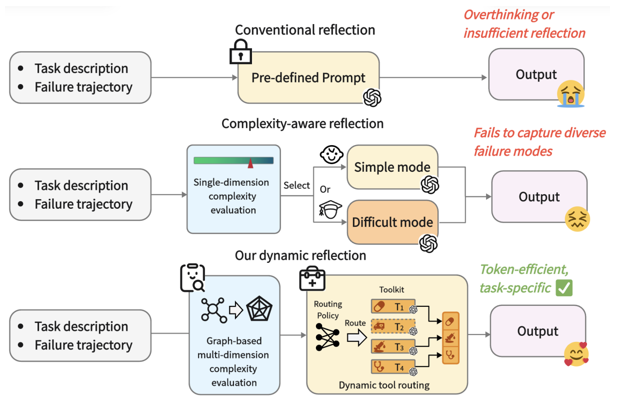

Conventional reflection
Overthinking or insufficient reflection
Task description
Failure trajectory
Pre-defined Prompt
Output
Complexity-aware reflection
Fails to capture diverse failure modes
Single-dimension complexity evaluation
Select
Or
Simple mode
Difficult mode
Our dynamic reflection
Token-efficient, task-specific
Graph-based multi-dimension complexity evaluation
Toolkit
Routing Policy
Route
T1
T2
T3
T4
Dynamic tool routing

Fig. 1. Compared with existing methods, DyRef dynamically evaluates task structures and failure trajectories using multi-dimensional complexity measures derived from a graph representation, and adaptively selects and routes appropriate reflection tools via a routing policy network.

as medicines or procedures—should be selected and combined. In a similar vein, effective reflection requires strategies that adaptively organize tools based on a multi-dimensional understanding of task complexity.

This raises two main challenges: **Lack of structured representation of task complexity.** Effective reflection depends on understanding why a task is complex—whether due to long horizons, spatial dispersion, or intricate object interactions. Such aspects are not independent; they are intertwined and often compound each other. Without a structured representation that disentangles these dimensions, existing methods resort to heuristic or uniform reflection, overlooking the actual sources of complexity. This mismatch makes reflection either too shallow to resolve real errors or unnecessarily heavy, wasting computational budget. **Besides, selecting the effective reflection strategy remains non-trivial.** The introduction of a multi-dimensional state representation creates a new problem: how to map this high-dimensional complexity to the optimal reflection strategy. This is a distinct challenge from prior heuristic selection rules, as it requires reasoning over a complex state space to balance efficacy and computational cost.

Instead of treating complexity as a flat set of statistics, we adopt a graph-based representation because tasks inherently involve heterogeneous entities (e.g., objects, actions, rooms) and multiple relation types (e.g., spatial, causal, functional). Graphs provide a natural and powerful abstraction to unify these elements, explicitly capturing dependencies, spatial arrangements, and interaction constraints—the very dimensions that constitute task complexity. This makes it possible to derive interpretable indicators of complexity along distinct dimensions from a single, structured representation.

Grounded in this intuition, we propose **DyRef**, a **Dy**namic **Ref**lection Framework via Graph-Based Complexity Measurement. First, we convert natural language task descriptions into a **Diagnostic Graph** that explicitly captures task goal, observation, and action trajectories through typed nodes and structural edges. From this graph we derive four **Complexity Factors**—Dependency, Spatial, Interaction, and Structure—which correspond to long horizons, wide spatial scope, reliance on appliance knowledge, and overall structure. Second, we introduce a **Routing Policy** network, trained in a self-supervised manner, that maps the factor vector to the most suitable tool from our **Reflection Toolkit**. Depending on the complexity, the Routing Policy adaptively assembles tools such as *Key Rule Extraction*, *Ambiguity Constraint*, *Experience Summary*, or *Lesson Pool*, forming a tailored reflection pipeline. By aligning reflection strategies with the diagnosed complexity, our method reduces reflection redundancy, prevents irrelevant error propagation, and improves overall planning success. Our contributions are threefold:

1) We provide a structured formalization of task complexity, introducing the Diagnostic Graph and deriving four Complexity Dependency, Spatial, Interaction, and Structure make task complexity explicit and computable.
2) To the best of our knowledge, this is the first reflection framework that dynamically adapts its strategies based on multi-dimensional complexity modeling, enabling robots to plan beyond static, heuristic-driven reflection.
3) We validate our approach through extensive experiments on simulated household benchmarks and real robotic platforms, showing that it significantly improves planning efficiency and success rates compared to uniform reflection baselines.

II. RELATED WORKS

A. LLMs for Task Planning

LLMs have been explored as planners for robotic tasks due to their reasoning and language understanding ability. Early work framed planning as sequence modeling with GPT-style policies [11], while ReAct [1] integrated reasoning with interaction. Other directions exploit program synthesis, generating code-like plans for greater reliability [12], [13], [14]. In embodied domains, methods such as SayCan [15] and LLM-Planner [16], [17] ground planning in robot affordances, ensuring that generated actions are executable. These studies show that LLMs can produce coherent plans, but they largely assume fixed reasoning capacity. When errors accumulate in long-horizon tasks, they lack mechanisms for explicit self-correction, motivating subsequent work on reflection.

B. Reflection for Robotic Task Planning

To address such errors, self-reflection has been introduced, where agents iteratively revise plans with feedback. Representative methods include Reflexion [4], Expel [5], and approaches that build structured knowledge [18] or combine reflection with search [19], [20], [21]. A newer line explores *complexity-aware reflection* [10], [8], [7], for example FCRF [7], which adapts reflection level based on task complexity. Yet such approaches typically rely on a single metric, limiting their ability to capture the multi-dimensional nature of task complexity. Thus, while reflection improves robustness, most methods lack a structured way to formalize complexity itself. This gap suggests the need for representations that can expose multi-dimensional task factors—a role naturally suited to graph structures.

C. Graph-based Representations for Task Reasoning

Graph structures have been widely explored to capture relational and compositional aspects of tasks. In embodied AI, scene graphs [22], [23] and object-centric graphs [24] encode entities and relations for affordance reasoning, while graph neural networks propagate dependencies across nodes to support long-horizon planning [25], [26]. These works highlight the value of structured representations for reasoning over dependencies. However, graphs are usually treated as static encodings or backbones, not as diagnostic tools that drive adaptive error correction. In contrast, our Diagnostic Graph derives explicit Complexity Factors and directly informs the dynamic selection of reflection modules, enabling complexity-aware adaptation in robotic planning.

In summary, LLMs have shown promise in task planning, reflection mechanisms enhance robustness, and graphs offer structured reasoning. However, prior work either treats task complexity implicitly or reduces it to a single metric, leaving no principled way to diagnose why tasks fail and how reflection should adapt. We address this gap by introducing a Diagnostic Graph that derives multi-dimensional Complexity Factors, enabling reflection to be both targeted and adaptive in robotic planning.

III. PRELIMINARIES

A. Planning Framework

A task can be described as a tuple $\langle \mathcal{G}, \mathcal{S}, \mathcal{O}, \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{G}$ denotes the task goal, $\mathcal{S}$ is the set of all possible states, $\mathcal{O}$ represents the observation space, $\mathcal{A}$ is the set of candidate actions, and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function that models state changes after executing an action. The aim of planning is to search for a decision policy, expressed as a sequence of actions, that drives the system from an initial state toward the desired goal state. In LLM-based planning practice, the above task information is typically provided through a natural language description D.

B. Self-Reflection Process in LLM-based Planning

Following Reflexion [4], we define Reflection as an iterative optimization process driven by environment feedback. At each trial t, the planner generates a trajectory τ_t through interaction with the environment, which returns a scalar outcome $r_t = M_{\mathcal{E}}(\tau_t)$. Reflection then transforms the pair (τ_t, r_t) into a verbal feedback summary $sr_t = M_{\text{ref}}(\tau_t, r_t)$, which is stored in memory `mem` to refine the decision policy in subsequent trials. The loop (τ_t, r_t, sr_t) continues until environment feedback indicates task success.

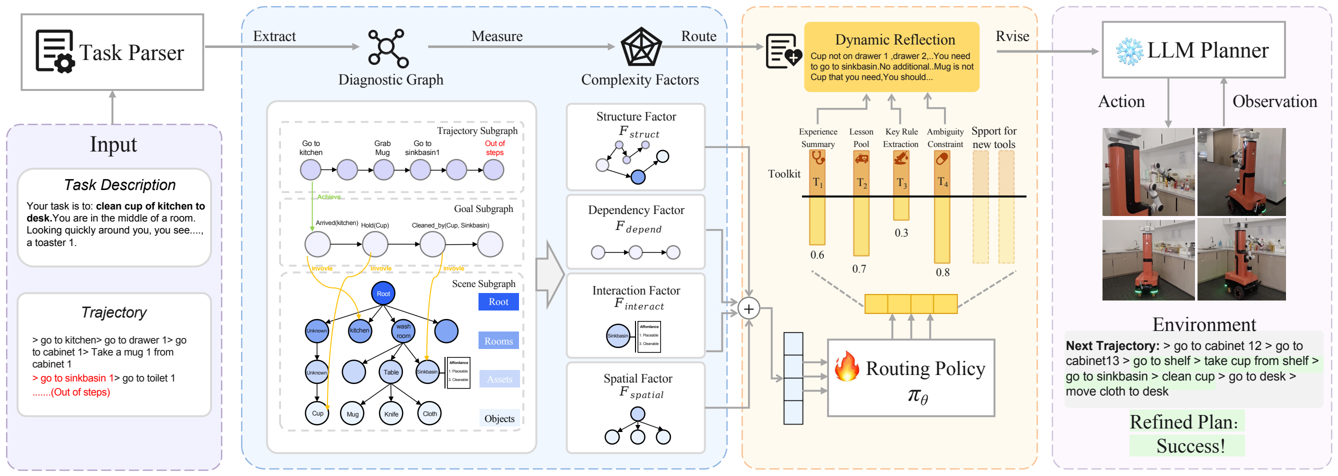

Task Parser
Input
Task Description
Your task is to: clean cup of kitchen to desk.You are in the middle of a room. Looking quickly around you, you see...., a toaster 1.
Trajectory
> go to kitchen> go to drawer 1> go to cabinet 1> Take a mug 1 from cabinet 1
> go to sinkbasin 1> go to toilet 1
.......(Out of steps)
Extract
Diagnostic Graph
Measure
Complexity Factors
Route
Trajectory Subgraph
Go to kitchen
Grab Mug
Go to sinkbasin1
Out of steps
Achieve
Goal Subgraph
Arrived(kitchen)
Hold(Cup)
Cleaned_by(Cup, Sinkbasin)
invovle
Scene Subgraph
Root
Unknown
kitchen
wash room
Table
Sinkbasin
Affordance
1. Placeable
3. Cleanable
Cup
Mug
Knife
Cloth
Rooms
Assets
Objects
Structure Factor F_{struct}
Dependency Factor F_{depend}
Interaction Factor $F_{interact}$
Spatial Factor $F_{spatial}$
Dynamic Reflection
Cup not on drawer 1 ,drawer 2,..You need to go to sinkbasin.No additional..Mug is not Cup that you need,You should...
Rvise
Experience Summary
Lesson Pool
Key Rule Extraction
Ambiguity Constraint
Spport for new tools
Toolkit
T_1
T_2
T_3
T_4
0.6
0.7
0.3
0.8
Routing Policy π_θ
LLM Planner
Action
Observation
Environment
Next Trajectory: > go to cabinet 12 > go to cabinet13 > go to shelf > take cup from shelf > go to sinkbasin > clean cup > go to desk > move cloth to desk
Refined Plan:
Success!

Fig. 2. Overview of the proposed DyRef framework. DyRef dynamically allocates reflection strategy according to task complexity. The pipeline consists of three components: (i) a hierarchical Diagnostic Graph that encodes task structure from descriptions and execution trajectories; (ii) four topological Complexity Factors (Dependency, Interaction, Spatial, Structure) derived from the graph, providing a structured representation of task complexity; (iii) a Routing Policy that leverages these factors to select tools from the Reflection Toolkit, enabling adaptive reflection and plan revision.

IV. METHODOLOGY

In this section, we detail our proposed framework, DyRef. As illustrated in Figure 2, DyRef integrates task representation, complexity measurement, and adaptive reflection into a unified pipeline. Specifically, it comprises three core components: (i) a hierarchical Diagnostic Graph that encodes task structure from descriptions and execution trajectories; (ii) four topological Complexity Factors extracted from the graph, which quantify different aspects of task complexity in an interpretable manner; (iii) a Routing Policy network that maps these factors to tools in the Reflection Toolkit, enabling tailored reflection strategies and efficient plan revision.

A. Diagnostic Graph

Reflection requires structured reasoning, while text-only representations obscure relations among actions, goals, and states. We therefore introduce the Diagnostic Graph G_t, a directed typed multigraph, constructed by a deterministic parsing pipeline: $G_t = \texttt{Parse}(D, \tau_t, r_t)$, where D is the task description, τ_t is the execution trajectory, and r_t is the environment feedback. The `Parse` function is implemented through rule-based entity extraction and event matching, which map task texts into goal predicates, execution logs into action nodes with temporal edges, and environment observations into scene nodes with spatial/functional relations.

The resulting graph is decomposed into three coupled subgraphs: (1) the *Trajectory Subgraph* G_t^{traj}, where nodes are executed actions and edges encode their temporal order as well as causal failures; (2) the *Goal Subgraph* G_t^{goal}, where nodes denote logical predicates and edges capture their dependency relations; and (3) the *Scene Subgraph* G_t^{scene}, which hierarchically organizes rooms, assets, and objects, with edges encoding spatial containment (room–asset–object), and cross-links to the goal nodes when entities are involved in predicates. Each node additionally carries attributes such as state (e.g., on/off), affordances (e.g., openable, heatable).

By jointly modeling temporal order, logical dependencies, spatial hierarchy, and functional relations, these subgraphs capture the core dimensions of task complexity—temporal dependencies, spatial arrangements, and object interactions—forming a grounded basis for adaptive reflection.

B. Graph-based Complexity Factors

Based on Diagnostic Graph, we extract four interpretable Complexity Factors, aligned with classical graph-theoretic descriptors: path-based, coverage-based, neighborhood-based, and high-order structural features. These factors provide simple yet interpretable signals of task complexity while remaining computationally tractable.

a) Dependency factor F_{depend}: This factor corresponds to **path-based complexity**, measuring the length of the longest dependency chain in the goal subgraph G_t^{goal}. A longer dependency path indicates deeper reasoning horizon and higher temporal complexity:

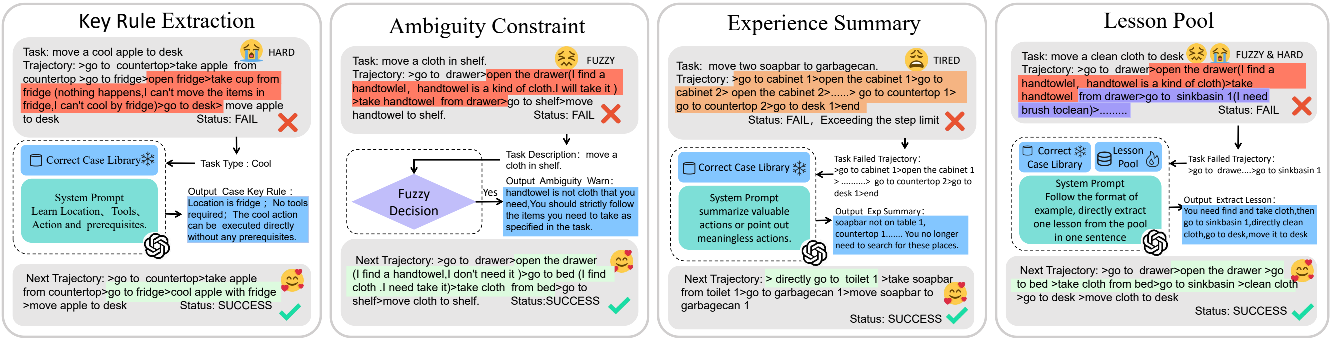
$$F_{\text{depend}} = \max_{p \in \mathcal{P}(G_t^{\text{goal}})} |p|, \tag{1}$$

where $\mathcal{P}(G_t^{\text{goal}})$ denotes the set of all dependency paths and $|p|$ their length.

b) Spatial factor $F_{spatial}$: This factor reflects **coverage complexity**, measuring the spatial spread of goal-relevant objects across locations in the scene subgraph G_t^{scene}. A wider spread increases the search overhead for navigation and object grounding:

$$F_{\text{spatial}} = \sum_{o \in V_{\text{goal}}^{\text{obj}}} |\text{Loc}(o)|, \tag{2}$$

where $\text{Loc}(o)$ is the set of possible asset locations associated with object o.


Key Rule Extraction
HARD
Correct Case Library
Task Type : Cool
System Prompt Learn Location、Tools、Action and prerequisites.
Ambiguity Constraint
FUZZY
Fuzzy Decision
Yes
Experience Summary
TIRED
Correct Case Library
System Prompt summarize valuable actions or point out meaningless actions.
Lesson Pool
FUZZY & HARD
Correct Case Library
Lesson Pool
System Prompt Follow the format of example, directly extract one lesson from the pool in one sentence

Task: move a cool apple to desk
Trajectory: >go to countertop>take apple from countertop >go to fridge>open fridge>take cup from fridge (nothing happens,I can't move the items in fridge,I can't cool by fridge)>go to desk> move apple to desk
Status: FAIL

Output Case Key Rule : Location is fridge ; No tools required; The cool action can be executed directly without any prerequisites.

Next Trajectory: >go to countertop>take apple from countertop>go to fridge>cool apple with fridge >move apple to desk
Status: SUCCESS

Task: move a cloth in shelf.
Trajectory: >go to drawer>open the drawer(I find a handtowlel, handtowel is a kind of cloth.I will take it) >take handtowel from drawer>go to shelf>move handtowel to shelf.
Status: FAIL

Task Description: move a cloth in shelf.

Output Ambiguity Warn: handtowel is not cloth that you need,You should strictly follow the items you need to take as specified in the task.

Next Trajectory: >go to drawer>open the drawer (I find a handtowel,I don't need it)>go to bed (I find cloth .I need take it)>take cloth from bed>go to shelf>move cloth to shelf.
Status:SUCCESS

Task: move two soapbar to garbagecan.
Trajectory: >go to cabinet 1>open the cabinet 1>go to cabinet 2> open the cabinet 2>......> go to countertop 1> go to countertop 2>go to desk 1>end
Status: FAIL, Exceeding the step limit

Task Failed Trajectory: >go to cabinet 1>open the cabinet 1 > ....> go to countertop 2>go to desk 1>end

Output Exp Summary: soapbar not on table 1, countertop 1....... You no longer need to search for these places.

Next Trajectory: > directly go to toilet 1 >take soapbar from toilet 1>go to garbagecan 1>move soapbar to garbagecan 1
Status: SUCCESS

Task: move a clean cloth to desk
Trajectory: >go to drawer>open the drawer(I find a handtowlel, handtowel is a kind of cloth)>take handtowel from drawer>go to sinkbasin 1(I need brush toclean)>.........
Status: FAIL

Task Failed Trajectory: >go to drawe....>go to sinkbasin 1

Output Extract Lesson: You need find and take cloth,then go to sinkbasin 1,directly clean cloth,go to desk,move it to desk

Next Trajectory: >go to drawer>open the drawer >go to bed >take cloth from bed>go to sinkbasin >clean cloth >go to desk >move cloth to desk
Status: SUCCESS

Fig. 3. The four modules of the Reflection Toolkit: (a) Key Rule Extraction, (b) Ambiguity Constraint, (c) Experience Summary, and (d) Lesson Pool. Each panel illustrates input–output format, core logic, and a minimal example, showing how the toolkit supports adaptive correction for robotic task planning.

c) Interaction factor F_{inter}: This factor captures **neighborhood complexity**, reflecting the functional richness of assets required by the goal. It is approximated by the degree of asset nodes in terms of available affordances:

$$F_{\text{inter}} = \sum_{a \in V_{\text{goal}}^{\text{asset}}} |\text{Aff}(a)|, \tag{3}$$

where $\text{Aff}(a)$ denotes the set of affordances of asset a.

d) Structure factor F_{struct}: Beyond local descriptors, this factor encodes **high-order structural complexity** of the full diagnostic graph. We adopt Graph2Vec [27], which aggregates rooted subgraph features into a continuous embedding:

$$F_{\text{struct}} = \text{Graph2Vec}(G_t) \in R^d. \tag{4}$$

Finally, all factors are concatenated into a single representation of task complexity: $\mathbf{x}_t = [F_{\text{depend}}, F_{\text{spatial}}, F_{\text{inter}}, F_{\text{struct}}]$. The hand-crafted factors emphasize interpretable local topology, such as dependency depth, spatial span, appliance interactions. In contrast, Graph2Vec provides a global structure embedding of the entire Diagnostic Graph, capturing higher-order structures beyond manual design.

C. Reflection Toolkit

The Reflection Toolkit is a collection of modular functions $\mathbf{T} = \{T_1, T_2, \ldots, T_K\}$, each sharing a common interface: given the diagnostic graph, trajectory, and feedback, a tool outputs a reflection snippet from a particular perspective. It is worth noting that our focus is not on the design of individual tools but on the overall framework that integrates them.

At trial t, the routing policy $\pi_\theta(\mathbf{x}_t)$ selects one or multiple tools based on task complexity. The selected tool outputs are order-invariant and concatenated into the final reflection text:

$$sr_t = \text{Concat}\big(\{T(G_t, \tau_t, r_t, \texttt{mem}) \mid T \in \pi_\theta(\mathbf{x}_t)\}\big), \tag{5}$$

which is injected back to guide the next trial. In our implementation, $\mathbf{T}$ consists of four representative tools, each of which has been validated in previous studies [7]: (1) *Key Rule Extraction*, abstracting preconditions and execution rules; (2) *Ambiguity Constraint*, detecting and restricting invalid operations; (3) *Experience Summary*, retaining effective actions from past trajectories; (4) *Lesson Pool*, leveraging corrected cases for deeper reflection. Figure 3 illustrates these tools, and extended implementation notes are available on our project webpage.

D. Routing Policy network

A key challenge in reflection is mismatching strategies to task structure, which can lead to redundant effort and error accumulation. To address this, we design a Routing Policy π_θ parameterized by θ, which maps the task representation $\mathbf{x}_t$ into reflection decisions:

$$\pi_\theta(\mathbf{x}_t) = \text{Mask}\{\text{Head}(W_2\, \sigma(W_1\, \mathbf{x}_t))\}\,, \tag{6}$$

where W_1, W_2 are trainable weight matrices, σ is a nonlinearity (SiLU), $\text{Head}(\cdot)$ projects into multiple action heads, and $\text{Mask}(\cdot)$ applies feasibility constraints. The multi-head outputs specify reflection actions—binary switches or categorical choices—determining which tools are activated and how reflection budgets are allocated. At inference, feasibility masks suppress inconsistent or overly costly configurations under low-complexity.

E. Offline Self-Supervised Training

Direct supervision from human labels is undesirable since tool choices can be biased and expensive to annotate, while reinforcement learning requires costly online interactions with slow convergence. Instead, we adopt an offline self-supervised scheme in which the Routing Policy learns from pseudo-labels automatically derived from execution outcomes.

For each task, a small candidate set of reflection strategies $\mathcal{A}$ is sampled and executed, where each strategy $a \in \mathcal{A}$ produces a success indicator $S(a) \in \{0, 1\}$ and a token cost $\text{cost}(a) \in R^+$. A pseudo-label $a^\star$ is then selected by trading off accuracy and efficiency:

$$a^\star = \arg\max_{a \in \mathcal{A}} \big[S(a) - \alpha\, \text{cost}(a)\big], \tag{7}$$

where $\alpha > 0$ is a weight controlling the trade-off. The Routing Policy, parameterized by θ, is trained to predict $a^\star$ with a loss that combines classification and cost penalty:

$$\mathcal{L}(\theta) = \text{CE}\big(\pi_\theta, a^\star\big) + \eta\, E_{a \sim \pi_\theta}[\text{cost}(a)]\,, \tag{8}$$

where π_θ is the predicted distribution over strategies, CE is the cross-entropy loss, $E_{a \sim \pi_\theta}[\cdot]$ denotes expectation with

TABLE I

SUCCESS RATE OF DYREF AND BASELINES ACROSS VARIOUS ALFWORLD TASKS. OUR DYREF OUTPERFORMS OTHER METHODS.

Method	Success Rate(%)							
	Put	Clean	Heat	Cool	Examine	SR @1	SR @3	ALL SR
Model : GPT 4o Series								
Plan-Only	64.5	66.6	30.0	62.5	26.6	61.9	67.9	68.7
Reason-Only	86.3	87.5	77.1	93.3	58.3	80.6	82.8	82.8
Reason-Reflect	84.9	84.3	80.0	93.3	58.3	80.6	82.8	83.6
Reason-Flex Reflect	91.4	**90.0**	93.5	**100.0**	63.6	82.1	90.3	91.0
Ours	**100.0**	**90.0**	**95.7**	**100.0**	**83.3**	**85.8**	**93.3**	**94.8**
Model : Gemini 2.5 Series								
Plan-Only	82.9	71.0	82.6	90.5	66.7	73.9	76.9	79.1
Reason-Only	90.2	80.6	79.2	95.7	55.6	76.1	82.1	83.6
Reason-Reflect	97.6	80.6	78.3	90.5	66.7	76.1	81.3	85.1
Reason-Flex Reflect	**100.0**	87.1	60.9	90.5	88.9	78.4	85.8	87.3
Ours	**100.0**	**93.5**	**91.3**	**100.0**	**83.3**	**91.0**	**93.3**	**94.8**

TABLE II

FLEXIBILITY AND EFFICIENCY OF DIFFERENT REFLECTION METHODS. OUR DYRE DEMONSTRATES SIGNIFICANTLY HIGHER FLEXIBILITY AND EFFICIENCY.

Methods	Efficiency				Flexibility	
	$\mathrm{Cost}_{\mathrm{ref}}$ @1 $\downarrow$	$\mathrm{Cost}_{\mathrm{ref}}$ @4 $\downarrow$	$\mathrm{Rate}_{\mathrm{rev}}(\%)$ @1 $\uparrow$	$\mathrm{Rate}_{\mathrm{rev}}(\%)$ @4 $\uparrow$	R $\uparrow$	CV(%) $\uparrow$
Model : GPT 4o Series						
Reason-Reflect	480.3	953.6	38.0	47.6	175	22.4
Reason-Flex Reflect	453.1	570.4	42.8	71.4	197	18.2
Ours	**260.3**	**408.8**	**66.1**	**87.5**	**268**	**25.4**
Model : Gemini 2.5 Series						
Reason-Reflect	909.0	913.7	42.9	64.3	207	23.5
Reason-Flex Reflect	366.3	558.3	48.2	69.6	237	21.5
Ours	**219.6**	**325.1**	**76.8**	**87.5**	**258**	**31.4**

respect to π_θ, and $\eta > 0$ is a coefficient that weights the expected cost term. This objective enables the Routing Policy to adaptively balance task success and reflection cost from data without additional supervision.

V. EXPERIMENTS

In this section, our framework is evaluated through experiments conducted in the common household environment of AlfWorld [28]. Furthermore, real-world robotic experiments are performed to validate the practicability of our method in physical environments. The proposed approach demonstrates significant advantages in both overall performance and specific metrics.

A. *Experimental Setup*

Environment and Dataset. We conduct our evaluation in **AlfWorld**, a text-based virtual household environment that includes five task categories: Put, Clean, Heat, Cool and Examine. Our experiments cover the full dataset of 134 tasks across these categories, each performed over five epochs of planning trials to ensure robust and generalizable results.

Compared methods. We benchmark our approach against four categories of representative baselines in robotic planning and reflection: 1. **Plan-Only**: ProgPrompt [13], which directly generates action sequences from textual task descriptions based on the principle of In-Context Learning [29] principle. 2. **Reason-Only**: ReAct [1], which integrates reasoning with action generation by using LLMs to iteratively decide the next step. 3. **Reason-Reflect**: Reflexion [4], which extends ReAct by enabling reflection on failures through environmental feedback, thereby improving subsequent planning. 4. **Reason-Flex Reflect**: FCRF [7], which integrates valuable historical experiences and lessons learned from failures, enables LLMs to flexibly self-reflect based on task complexity.

Metrics. We evaluate the methods from three aspects:

- **Success Rate** measures the overall effectiveness of reflection and planning by calculating the proportion of tasks successfully completed at the end of the trial.
- **Efficiency** is assessed through two indicators:
  - **Reflection Cost** ($\mathrm{Cost}_{\mathrm{ref}}$): the average number of tokens consumed during reflection, defined as $\mathrm{Cost}_{\mathrm{ref}} = \frac{\mathrm{Cost}_{\mathrm{all}}}{\mathrm{E}_{\mathrm{corrected}}}$, Where $\mathrm{Cost}_{\mathrm{all}}$ denotes the total number of tokens consumed across all experimental rounds, and $\mathrm{E}_{\mathrm{corrected}}$ denotes the number

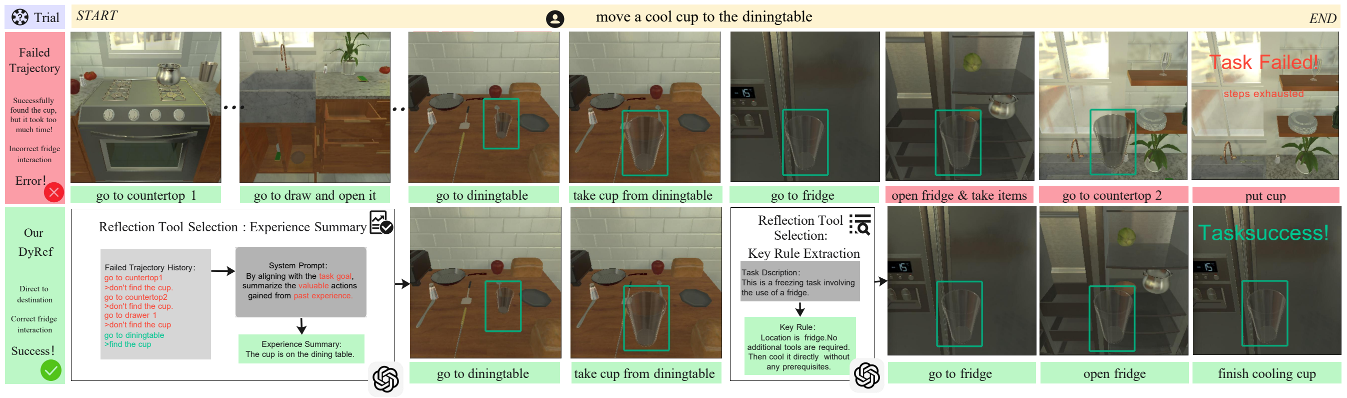

Trial
START
move a cool cup to the diningtable
END
Failed Trajectory
Successfully found the cup, but it took too much time!
Incorrect fridge interaction
Error!
go to countertop 1
go to draw and open it
go to diningtable
take cup from diningtable
go to fridge
open fridge & take items
go to countertop 2
put cup
Task Failed!
steps exhausted
Our DyRef
Direct to destination
Correct fridge interaction
Success!
Reflection Tool Selection : Experience Summary
Failed Trajectory History:
go to cuntertop1
>don't find the cup.
go to countertop2
>don't find the cup.
go to drawer 1
>don't find the cup
go to diningtable
>find the cup
System Prompt:
By aligning with the task goal, summarize the valuable actions gained from past experience.
Experience Summary:
The cup is on the dining table.
go to diningtable
take cup from diningtable
Reflection Tool Selection:
Key Rule Extraction
Task Dscription:
This is a freezing task involving the use of a fridge.
Key Rule:
Location is fridge.No additional tools are required. Then cool it directly without any prerequisites.
go to fridge
open fridge
finish cooling cup
Tasksuccess!

Fig. 4. In an AlfWorld example, DyRef corrects a failed trajectory by selecting a combined reflection strategy of experience summarization and key rule extraction, utilizing task, environmental, and historical execution data.

of tasks corrected through reflection across all episodes.

  - **Revision Rate** ($\mathrm{Rate}_{\mathrm{rev}}$): the proportion of errors corrected through reflection, defined as $\mathrm{Rate}_{\mathrm{rev}} = \frac{\mathrm{E}_{\mathrm{corrected}}}{\mathrm{E}_{\mathrm{initial}}}$,Where $\mathrm{E}_{\mathrm{initial}}$ represents the number of tasks that failed during the initial planning phase.

- **Flexibility** evaluates the adaptability of reflection length to task complexity, quantified by:
  - **Range** (R): The difference between the maximum number of tokens in a single reflection and the minimum number of tokens in a single reflection.
  - **Coefficient of Variation** (CV): normalized dispersion, defined as $\mathrm{CV} = \frac{\mathrm{Cost}_{\mathrm{ref,STD}}}{\mathrm{Cost}_{\mathrm{ref,AVE}}}$

B. *Results*

The main results are presented in Table I and Table II. To balance performance and computational cost, we use the official *GPT-4o Series* and *Gemini 2.5 Series* models in all experiments. Main results show: 1. Our DyRef achieved a 100% success rate in both the Put and Cool tasks. Compared to the baseline methods, its success rate improved by over 30% in examine tasks and by over 4% in the overall performance, demonstrating a significantly enhanced reflexion capability, as detailed in Table I. This improvement stemmed from its mechanism of analyzing execution failures, dynamically selecting reflection strategies based on task complexity, and effectively integrating insights from both successful and failed trajectories. Figure 4 illustrates an application of DyRef to a long-term task. 2. No significant correlation was observed between the length of reflection-generated tokens and their subsequent error-correction capability. Furthermore, empirical evidence indicated that compromised self-reflection flexibility and efficiency were associated with diminished success rates. These findings underscore the necessity of our research focus and validate the rationality of the metrics defined in our study. 3. Among all evaluated methods, the **Plan-Only** approach, which utilized simple planning based solely on input and context, achieved the lowest success rate. The **Reason-Only** and **Reason-Reflect** methods performed better. Although **Reason-Reflect** generally outperformed **Reason-Only**, it underperformed in certain specific tasks. This result suggests that self-reflection with fixed templates does not invariably lead to performance improvement. 4. In contrast to previous methods, DyRef dynamically adjusted the volume of reflection across a wider range with higher variability, as shown in Table II. The value of R increased by over 8.9%, and the value of CV rose by more than 39.6%, demonstrating its superior **flexibility**. DyRef also exhibited a higher revision rate and a lower reflection cost in the **efficiency** metrics, indicating robust error-correction capability and cost-effectiveness. Specifically, the $\mathrm{Rate}_{\mathrm{rev}}$ increased by more than 22.5%, while the $\mathrm{Cost}_{\mathrm{ref}}$ decreased by more than 28.4%. These results show that DyRef adaptively allocated computational resources, effectively learned from all execution outcomes, and achieved excellent overall cost-effectiveness.

VI. ANALYSIS AND DISCUSSION

A. *Episode Analysis of Self-Reflection Process*

We conducted an episode analysis to investigate the mechanisms and manifestations of the self-reflection process (Figure 6). The LLM was instructed to replan erroneous long-horizon tasks across five experimental rounds, and the completion status was observed in each episode. Key observations included the following: 1. As shown in the left figure, DyRef outperformed all baseline methods in every episode. The average revision rate increased by at least 22.1%, while reflection costs decreased by at least 38.1%. These results demonstrated the superior performance, robustness, and computational efficiency of our method. 2. Across all episodes, the Plan-Only method exhibited the lowest error-correction performance. Reason-Only and Reason-Reflect performed better yet showed similar results to each other. Reason-Flex Reflect significantly improved error correction compared to previous methods, although minimal improvement was observed in the first reflection round, indicating that it still had substantial room for overall enhancement. Our DyRef achieved a 38.6% improvement in first-round error-correction accuracy. These findings indicated that inflexible reflection hindered error correction and increased cost. In contrast, our

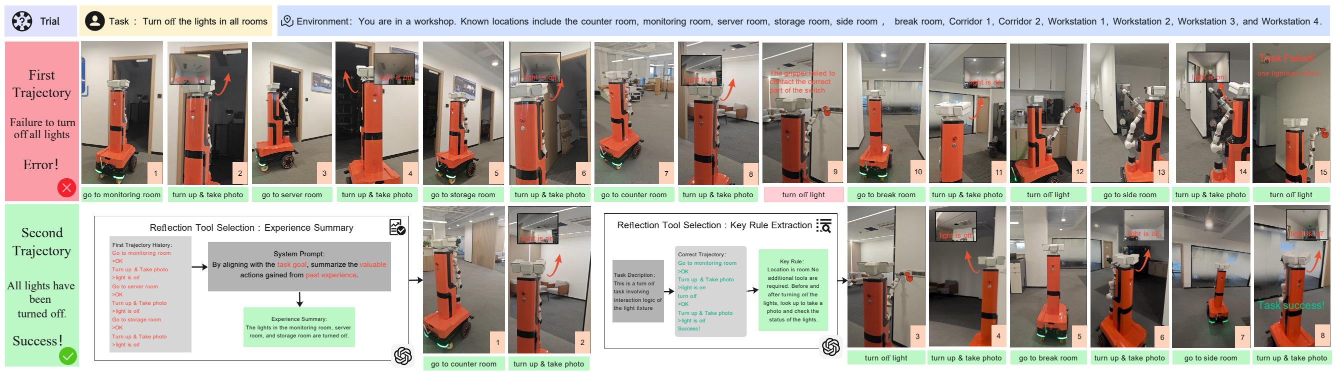

Trial
Task : Turn off the lights in all rooms
Environment: You are in a workshop. Known locations include the counter room, monitoring room, server room, storage room, side room , break room, Corridor 1, Corridor 2, Workstation 1, Workstation 2, Workstation 3, and Workstation 4.
First Trajectory
Failure to turn off all lights
Error!
go to monitoring room
turn up & take photo
go to server room
turn up & take photo
go to storage room
turn up & take photo
go to counter room
turn up & take photo
turn off light
go to break room
turn up & take photo
turn off light
go to side room
turn up & take photo
turn off light
The gripper failed to contact the correct part of the switch
Task Failed!
Second Trajectory
All lights have been turned off.
Success!
Reflection Tool Selection : Experience Summary
First Trajectory History: Go to monitoring room >OK Turn up & Take photo >light is off Go to server room >OK Turn up & Take photo >light is off Go to storage room >OK Turn up & Take photo >light is off
System Prompt: By aligning with the task goal, summarize the valuable actions gained from past experience.
Experience Summary: The lights in the monitoring room, server room, and storage room are turned off.
go to counter room
turn up & take photo
Reflection Tool Selection : Key Rule Extraction
Task Dscription: This is a turn off task involving interaction logic of the light fixture
Correct Trajectory: Go to monitoring room >OK Turn up & Take photo >light is on turn off >OK Turn up & Take photo >light is off Success!
Key Rule: Location is room.No additional tools are required. Before and after turning off the lights, look up to take a photo and check the status of the lights.
turn off light
turn up & take photo
go to break room
turn up & take photo
go to side room
turn up & take photo
Task success!

Fig. 5. Example of a failed task successfully refined using our framework.

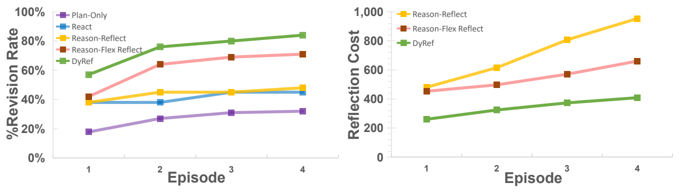


Fig. 6. Result of revision and reflection cost with different episodes. Our DyRef outperforms in all episodes, converges more quickly and has a significant lower cost.

TABLE III
ABLATION OF MODULES OF DYREF IN ALFWORLD TASKS. ALL DESIGNED HIERARCHICAL GRAPH CONTRIBUTE TO THE PERFORMANCE.

Method	**SR** (%)		**R** ↑	**CV** ↑	**Cost**$_{\textbf{ref}}$ ↓	**Rate**$_{\textbf{rev}}$ ↑
	SR@1	**SR@All**				
Ours full	**85.8**	**94.8**	268	**25.4**	408.8	**87.5**
w/o Struct	83.6	89.6	198	24.7	**360.7**	75.0
w/o Depend	78.4	82.9	208	25.1	775.3	58.9
w/o Interact	72.3	83.6	**272**	23.6	823.2	60.7
w/o Spatial	76.9	85.1	258	23.6	810.4	64.3

DyRef effectively utilized examples and lessons derived from failures to achieve more effective and cost-efficient reflection.

B. Ablation Study

To demonstrate the effectiveness of each feature and explore their interrelationships, we conducted ablation studies on the AlfWorld dataset, with results summarized in Table III. In the **w/o Dependence Factor** configuration, the overall revision rate was reduced by 48.6%, the success rate decreased by 11.9%, and the R value dropped by 28.8% due to the neglect of task sub-objectives. Under the **w/o Spatial Factor** setting, the reflection coefficient of variation decreased by 7.6% as a result of unaccounted environmental complexity. For the **w/o Interaction Factor** condition, the first-trial success rate declined by 18.7%, while the revision cost increased by 50.3% owing to ignored interaction logic complexity. In the **w/o Structure Factor** scenario, where the complexity of the diagnostic graph was disregarded, reflection costs were reduced by 11.8% compared to the complete model. However, the R value decreased by 26.1% and the revision rate dropped by 14.3%, indicating a trade-off in reflection flexibility and efficiency. The efficiency and flexibility of the method decreased when each feature was ablated, leading to a reduction in the overall success rate. Overall, the results of the ablation study met expectations across all metrics, demonstrating the contribution of each feature within our framework and confirming the rationality of our feature selection.

C. Real-World Robotic Experiment

We developed a plan-reflect loop system based on React-DyRef to validate its practical application potential. A robot equipped with a robotic arm and a binocular camera system was employed for the experiments. The robotic arm featured parallel grippers and a wrist-mounted camera, and was mounted on a mobile base. The implemented functions included Put(object, location), Grab(object), Move_to(location), Take_picture(), Turn_off(), and Turn_on(). The robot's task execution was designed with a maximum of 10 steps per task.

The system was tested on complex long-sequence tasks in real-world scenarios, such as turning off all room lights. The robot was required to inspect the status of all room lights on the map and ensure they remained off. Through multiple test iterations, our framework demonstrated high reliability and the capability to correct errors swiftly and accurately. After 20 repeated experiments, the first-pass reflection correction rate reached 90%. Additional details and demonstrations of the real-world scenario experiments are available on our project website and in supplementary videos.

VII. CONCLUSIONS

We presented **DyRef**, a dynamic reflection framework that diagnoses task complexity through a Diagnostic Graph and routes tasks to tailored strategies from a Reflection Toolkit. This design moves beyond fixed heuristics by aligning reflection level with the structural demands of each task. Experiments in AlfWorld and on real robots show that DyRef improves planning efficiency, raising success rates while reducing redundant reflections. These results highlight the importance of complexity-aware reflection in scaling LLM-based planners to long-horizon tasks.

REFERENCES

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[2] Jiatao Zhang, Lanling Tang, Yufan Song, Qiwei Meng, Haofu Qian, Jun Shao, Wei Song, Shiqiang Zhu, and Jason Gu. Fltrnn: Faithful long-horizon task planning for robotics with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6680–6686. IEEE, 2024.

[3] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

[4] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:8634–8652, 2023.

[5] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 19632–19642, 2024.

[6] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

[7] Yufan Song, Jiatao Zhang, Zeng Gu, Qingmiao Liang, Tuocheng Hu, Wei Song, and Shiqiang Zhu. Fcrf: Flexible constructivism reflection for long-horizon robotic task planning with large language models. *arXiv preprint arXiv:2507.14975*, 2025.

[8] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*, 2025.

[9] Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning (CoRL)*, pages 3468–3484. PMLR, 2023.

[10] Zehui Ling, Deshu Chen, Hongwei Zhang, Yifeng Jiao, Xin Guo, and Yuan Cheng. Fast on the easy, deep on the hard: Efficient reasoning via powered length penalty. *arXiv preprint arXiv:2506.10446*, 2025.

[11] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:31199–31212, 2022.

[12] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[13] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[14] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:58202–58245, 2024.

[15] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[16] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2998–3009, 2023.

[17] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application. *IEEE Access*, 2023.

[18] Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Generating instruction manuals by llm agents via interactive environmental learning. *arXiv preprint arXiv:2405.16247*, 2024.

[19] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:11809–11822, 2023.

[20] Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*, 2024.

[21] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

[22] Saeid Amiri, Kishan Chandan, and Shiqi Zhang. Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters*, 7(2):5560–5567, 2022.

[23] Christopher Agia, Krishna Murthy Jatavallabhula, Mohamed Khodeir, Ondrej Miksik, Vibhav Vineet, Mustafa Mukadam, Liam Paull, and Florian Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning (CoRL)*, pages 46–58. PMLR, 2022.

[24] Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. IEEE, 2024.

[25] Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 35, pages 11962–11971, 2021.

[26] Ruidong Ma, Yanan Liu, Erich W Graf, and John Oyekan. Applying vision-guided graph neural networks for adaptive task planning in dynamic human robot collaborative scenarios. *Advanced Robotics (AR)*, 38(23):1690–1709, 2024.

[27] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[28] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[29] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.