ControlLLM: Augment Language Models with Tools by Searching on Graphs

Zhaoyang Liu[1,2†], Zeqiang Lai[2†], Zhangwei Gao[2,3†], Erfei Cui[2,3†], Ziheng Li[4], Xizhou Zhu[4,2], Lewei Lu[5], Qifeng Chen[1✉], Yu Qiao[2], Jifeng Dai[4,6], and Wenhai Wang[7,2✉]

[1]HKUST [2]OpenGVLab, Shanghai AI Laboratory
[3]Shanghai Jiao Tong University [4]Tsinghua University [5]SenseTime
[6]BNRist [7]The Chinese University of Hong Kong

Abstract. We present ControlLLM, a novel framework that enables large language models (LLMs) to utilize multi-modal tools for solving complex real-world tasks. Despite the remarkable performance of LLMs, they still struggle with tool invocation due to ambiguous user prompts, inaccurate tool selection and mismatched input arguments. To overcome these challenges, our framework comprises three key components: (1) a *task decomposer* that breaks down a complex task into clear subtasks with well-defined inputs and outputs; (2) a *Thoughts-on-Graph (ToG) paradigm* that searches the optimal solution path on a pre-built tool graph, which specifies the parameter and dependency relations among different tools; and (3) an *execution engine with a rich toolbox* that interprets the solution path and runs the tools efficiently on different computational devices. We evaluate our framework on diverse tasks involving image, audio, and video processing, demonstrating its superior accuracy, efficiency, and versatility compared to existing methods. The code is available at `https://github.com/OpenGVLab/ControlLLM`.

Keywords: Large language models · Tool-augmented LLMs · ControlLLM · Multi-modal Interaction

1 Introduction

Large-scale language models, such as ChatGPT [28] and LLaMA series [40, 41], have demonstrated impressive capability in understanding and generating natural language. Beyond their prowess in linguistic tasks, these models have been rapidly extended to interaction, planning, and reasoning, propelling the advancement of studies in multi-modal interaction [1, 19, 20, 25, 42, 43, 55].

One of the emerging examples of multi-modal interaction is tool-augmented language models [24, 35, 36, 47, 49], which strive to enhance the capabilities of

† This work is done when they are interns at Shanghai AI Laboratory.
✉ Corresponding authors (wangwenhai@pjlab.org.cn, cqf@ust.hk).

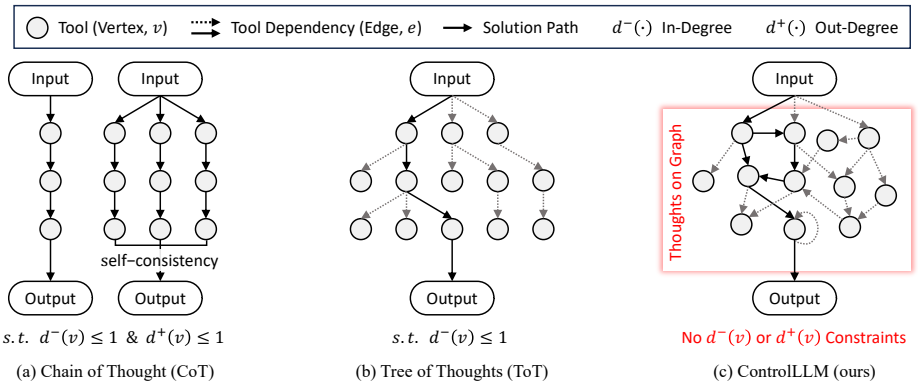

Fig. 1: Comparisons of different paradigms for task planning. (a) Chain of Thought (CoT) [45], CoT with self-consistency [44] and (b) Tree of Thoughts [50] (ToT) essentially rely on the LLMs to perform task planning, where the edge is actually formed by LLMs at run time. (c) The Thoughts-on-Graph (ToG) paradigm in our method searches for solutions on a pre-built graph that captures the dependencies of tools, which avoids the hallucination problem in tool invocation.

language models to include diverse modalities beyond text such as image, video, audio, *etc*. These models employ LLMs as primary controllers and incorporate tools with diverse functionalities as plugins, which solves a wide range of multi-modal tasks. However, challenges in this field still persist, covering task decomposition, task planning, and efficient tool scheduling.

With these challenges in mind, prior methods [24, 33, 36, 39, 47, 49, 51] made their endeavors in developing tool-augmented LLMs. They utilize LLMs with input-output prompting, CoT [45] or ToT [50] to perform task planning. These methods can solve problems by breaking them into a chain or tree of sub-tasks. Theoretically, as long as LLMs have strong generalization ability, these methods can also solve complex tasks. However, in practice, we found that these methods often suffer from inaccurate tool invocation problems when dealing with complex cases. This is due to the fact that solutions for complex tasks often contain tool invocations with intricate topological structures. It is insufficient for these methods to form a complex thought network and thus fail to solve complicated tasks. Therefore, it requires us to figure out a new paradigm beyond chain-shaped or tree-shaped ones, which can generate solutions with intricate topology structures to solve more complicated problems (see Fig. 1 and Fig. 2).

To this end, we introduce ControlLLM, a new framework that assists large language models in accurately and efficiently controlling multi-modal tools and identifying comprehensive solutions for complex real-world tasks involving multi-modal inputs. Alongside a variety of improvements over previous works, our framework places particular emphasis on three aspects as follows:

Task Decomposition. A task decomposer is introduced to analyze the user prompt and breaks it down into a number of subtasks, each with well-defined

attributes such as task description, task domain, arguments, and returned output. By decomposing complex tasks into manageable subtasks, the task decomposer significantly enhances the system's ability to handle intricate user prompts, which paves the way for follow-up task planning and solution execution.

Task Planning. This part handles tool selection and tool argument assignment. We propose a thoughts-on-graph (ToG) paradigm that traverses a topological tool graph to search for solutions. The nodes of the graph are tools that are connected based on their dependencies and relationships. ToG orchestrates the selected tools and controls the flow of resources among them to form possible solutions. ToG can find the optimal solution for each sub-task by applying diverse search strategies on the graph. Due to the concrete definition in subtask and explicit tool dependencies in a tool graph, ToG can effectively search all feasible solution paths in cases where the selected optimal solution fails to meet users' preferences.

Solution Execution. We design an execution engine that can execute the solution generated by ToG and craft informative and well-formatted responses. The engine has access to a versatile toolbox consisting of various tools from different sources, *e.g.*, locally deployed APIs or cloud services. The engine can also parallelize the tool executions according to the topology of the solution path to reduce the latency and provide feedback during the execution.

Our ControlLLM offers several advantages. (1) It can accurately handle complex real-world tasks that involve multi-modal inputs and outputs, while previous methods [4,22,24,36,47,49] usually fail to handle due to their capabilities of task planning; (2) It can overcome the token limitation of LLMs during task planning. Because our method searches the optimal solution path on the tool graph, instead of asking LLMs to generate a solution for the task; (3) It can easily scale up toolbox. Since all solutions lie in the tool graph, when tools change, we only need to rebuild the graph without re-training LLMs or updating in-context prompts.

To evaluate the effectiveness of ControlLLM in tasks of different complexities, we construct a benchmark with a series of tailored metrics. Specifically, we use irrelevant tool inclusion rate and necessary tool inclusion rate to measure tool selection. We employ the resource hallucination rate and resource type consistency rate to assess argument assignments. We also split the test set into three difficulty levels based on the number of APIs involved: easy (< 2 APIs), medium (2 or 3 APIs), and hard (> 3 APIs). We conducted various experiments, both quantitatively and qualitatively, to compare our method with existing ones. The results show that ControlLLM achieves a higher success rate in tool invocation, especially for complicated instructions.

In summary, the main contributions are as follows:

(1) We design three tailored components in this paper: Task decomposition, which breaks down the user prompt into subtasks with well-defined inputs and outputs; ToG paradigm for task planning, searching the optimal solution path on a graph that depicts tool dependencies; And an execution engine with a powerful toolbox, which efficiently schedules and executes the solution.

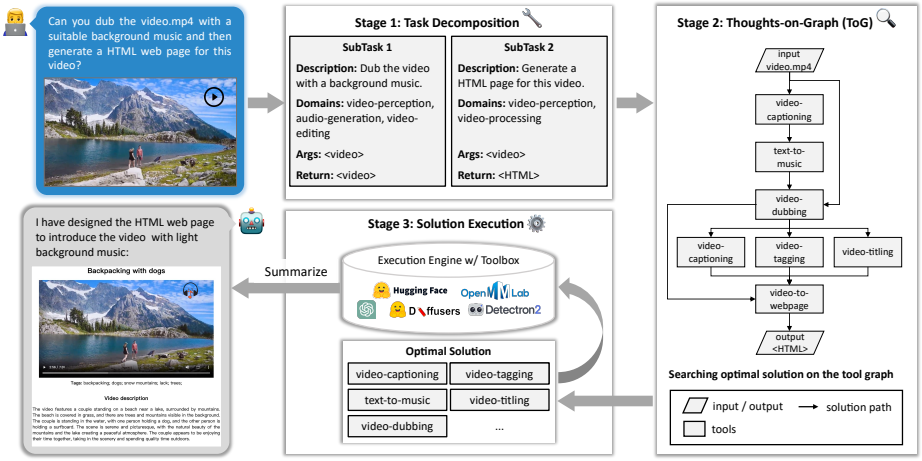

Fig. 2: System design of ControlLLM. The framework consists of three stages. The first stage is task decomposition, which parses the user input into several subtasks. Then, in Stage 2, ToG utilizes a depth-first search algorithm to find the optimal solution for each subtask. The execution engine in the last stage executes the solution and returns the output to users. We here use the example of generating a web page for the video to illustrate our method.

(2) We propose a framework, termed ControlLLM, that lets LLMs use various tools across different modalities to solve complex tasks in the real world. Our system has greatly improved the current state of the art in this field, and the system's capability has gone from unusable to basically usable. And we also extend the test data from toy cases to more complex problems in reality.

(3) We construct a benchmark to assess the efficacy of ControlLLM on tasks with different complexity levels. The experimental results demonstrate significant improvements in tool usage. Notably, ControlLLM achieves a success rate of 93% in the metric of overall solution evaluation, while the best baseline only reaches 59%. It lays a solid foundation for future LLMs-based system call.

2 Related Work

Planning, Reasoning, and Decision Making. It is a longstanding vision to empower autonomous agents with the abilities of planning, reasoning, and decision-making [18, 37, 46]. Despite progressive development, it was recent advancements in large language models (LLM) [3, 5, 29, 40, 54] that have taken a breakthrough step in addressing these problems on the broad user requests. Nevertheless, it is shown that LLMs still suffer from difficulties in dealing with knowledge-heavy and complex tasks [34]. To overcome these issues, Chain of Thoughts (CoT) [45] is introduced as a simple Tool Documentation Enables Zero-Shot Tool-Usage with Large Language Modelsprompting technique to elite

the complex reasoning capabilities of LLMs. Following this line of work, CoT with self consistency [44], Tree of Thoughts (ToT) [45], and other techniques [6,14,59], have been proposed to improve the reasoning abilities further. There are also several works [2, 52] that introduce techniques called Graph-of-Thought (GoT). They all share a common insight that relies on LLMs to generate thoughts for solving complicated NLP problems. In contrast, our ToG aims to endow the language model with the ability to use tools for a multi-modal dialogue system. Furthermore, ToG builds a tool graph in advance without requiring LLMs and uses a search algorithm to form a complicated thought network for task planning.

Tool-Augmented LLM. Drawing inspiration from the evolving planning and decision-making capabilities observed in Large Language Model (LLM) systems, a new wave of research starts to enhance LLMs with external tools for accessing up-to-date information, reducing hallucination, multi-modal interactions, *etc*. Prominent examples include ReAct [51], VisProg [11], ViperGPT [38], Visual ChatGPT [47], HuggingGPT [36], InternGPT [24], AutoGPT[1], and Transformers Agent[2]. A distinctive trait of this line of research is its reliance on the zero-shot or few-shot in-context learning [8] capabilities inherent in LLMs [3]. These capabilities enable task decomposition, tool selection, and parameter completion without requiring explicit finetuning. However, due to the inherent limitations of LLMs, issues such as hallucination and challenges in effective decomposition and deduction can arise with substantial frequency. Furthermore, there are also instruction-tuning methods [10,12,30,31,33,35,49]. Whereas alleviating the above issues after being tuned on the text corpus involved tools, these methods are still limited at expanding the toolset, i.e., additional training is required to add tools. Among these methods, ToolLLM [33] proposes the depth-first search-based decision tree to boost the planning ability of LLMs.

Multi-Modal LLMs. Developing LLMs with multi-modal capabilities is another approach to deal with more complex real-world scenarios [7, 13, 15, 16, 21–23, 26, 27, 32, 48]. For instance, BLIP-2 [19], LLava [23], and Mini-GPT4 [57] bind frozen image encoders and LLMs to enable the vision-language understanding and generation. Similarly, VisionLLM [43] and LISA [17] empower the LLMs with the visual perception capabilities such as object detection and segmentation. These works [9, 16, 58] extend LLM for interleaved image and text generation by jointly optimizing the LLM with off-the-shelf Stable Diffusion model. Kosmos2 [32], Ferret [53], GPT4RoI [56], and *etc*., design various region-aware image encoders to augment LLMs with the abilities of grounding and referring. Nevertheless, these methods could only cover a limited range of modalities or tasks and often require huge effects on model finetuning.

3 ControlLLM

The prevalence of LLMs has unprecedentedly boosted the development of human-computer interaction. It is feasible to empower the LLMs with abilities to in-

[1] `https://github.com/Significant-Gravitas/Auto-GPT`
[2] `https://huggingface.co/docs/transformers/transformers_agents`

Table 1: The output protocol of task decomposition. We elaborate on each field in the output of task decomposition.

Field	Description
description	a brief summary of subtask. It gives some guidance on how to approach the problem for ToG.
domains	the domain scope that tools required by this task fall into. It helps ToG narrow down the search space and find the most relevant and suitable tools for the subtask. We showcase the domains in the appendix.
args	the inputs that the user provides for this subtask. It is usually in the form of key-value pairs, where the key is the type of the argument, and the value is the resource path or text you want to use. For example, [{"type": "image", "value": "image_1.png"}, {"type": "text", "value": "remove the dog in the picture"}].
return	the expected output of the subtask. For example, the return is {"image": "⟨GEN⟩-0"}, which means the expected output is an image. "⟨GEN⟩-0" is a temporary placeholder, which can be used to capture the dependency between different subtasks."

teract with various modalities via tools. In response, we present an innovative framework, namely **ControlLLM**, characterized by its flexibility and high performance. As depicted in Fig. 2, our framework consists of three sequential stages, *i.e.*, task decomposition, task planning and solution execution.

3.1 Task Decomposition

ControlLLM starts with task decomposition – a stage for decomposing the user request r into a list of subtasks. We here can utilize a language model $\mathcal{M}$, *e.g.*, ChatGPT or instruction-tuned LLaMA, to automatically decompose the user request as follows:

$$\{s_0, ..., s_i, ..., s_n\} = \mathcal{M}(r), \tag{1}$$

where s_i is the i-th subtask, n is the number of all subtasks. We will elaborate on the different choices of language model $\mathcal{M}$ in Sec. 3.4. The result of task decomposition is in JSON format, and the output protocol is presented in Table 1.

Task decomposition is different from task planning. It only breaks down the user's request into several subtasks and summarizes the input resources for each subtask from the user request. It does not need to know what tools to use or how to use them. The objective of this stage is to achieve three aims. Firstly, it splits user requests into smaller and more manageable units, *i.e.*, subtasks, thereby accelerating task planning. Secondly, it seeks to determine the task domain that is most relevant and appropriate for the given problem, thus further narrowing down the scope of task planning. Thirdly, it endeavors to infer the input and output resource types from the context, which identifies the start and end nodes for ToG to search in the next stage.

3.2 Task Planning with Thoughts-on-Graph

This stage is the key to the entire system. Given the results of task decomposition, we design a Thoughts-on-Graph (ToG) paradigm to find solutions on the graph heuristically.

Building the Tool Graph In this stage, we first construct a Tool Graph G using an adjacency matrix, which serves as a fundamental guideline for analyzing and optimizing the interactions between tools. Our motivation is driven by observing a discernible topological structure that inherently exists between the input and output of diverse tools, as demonstrated in Fig. 2. This compelling insight propels us to craft a comprehensive tool graph that holds the inherent relationship between tools.

There are two types of nodes *i.e.*, *Resource* node and *Tool* node, in the graph. *Resource* node can be formally defined as one-tuple: ⟨`type`⟩, where `type` shows the specific type of resource, like image, mask, video, *etc*. *Tool* node can be expressed as a three-tuple: ⟨`desc, args, ret`⟩. The `desc` field encapsulates the description of the tool, including its purpose and intended applications. The `args` field denotes a list of resource nodes that the tool accepts, thereby giving the prerequisites for utilizing this tool. Finally, the `ret` field designates the resource node that the tool returns.

Edge Definitions. Edges in the tool graph intricately connect the nodes, highlighting the relationships between different tools. We define two types of edges in the graph: (1) *Tool-resource edge* is established from the tool to its returned resource type. This signifies that the tool is capable of generating resources of the corresponding type. (2) *Resource-tool edge* denotes the resource node that can be accepted as input arguments for its adjacent tool. This connection indicates how the resources flow to the tool. Through the establishment of this graph, we can use diverse search strategies to make informed decisions regarding tool selection and input resource assignments.

Searching on the Graph As described in Algorithm 1, our ToG is built upon a depth-first search (*DFS*) algorithm where the tool selection function $\mathcal{F}$ is used to sample the tool nodes on the tool graph. ToG relies on the outputs of task decomposition in stage 1. The algorithm starts from the input resource nodes and explores all possible paths to the output node while keeping track of the intermediate resources and tools. The algorithm stops when it reaches the expected output node or when it exceeds a maximum length limit m (m=10 by default). Finally, ToG returns all searched solutions. Each step from *resource node* to *tool node* represents a thought process, as it involves a decision that determines whether to use this tool and how to assign its input arguments.

To find a trade-off between time and space complexities, we develop a tool assessment module in which we prompt the language model to score the tools in each search step and then filter out some irrelevant tools. As such, we design four search strategies for the function $\mathcal{F}$ to determine which tool nodes within the task domains to visit among all adjacent nodes when searching on the graph:

Algorithm 1 The pseudocode of depth-first search in Thoughts-on-Graph

```
Input:
    t: subtask obtained by Eq. 1 in Stage 1
    g: tool graph G constructed in Sec. 3.2
    r: available resources, initialized with subtask["args"]
    s: recorded tools during searching
Output:
    solutions: all possible solutions for the subtask t
function DFS_Search(t, g, r, s)
    if len(s) > m:
        return []
    # F finds all tool candidates, explained in Sec. 3.2
    available_tools = F(t, g, r)
    solutions = []
    for tool in available_tools:
        s.append(tool)
        r.append(tool["returns"])
        if tool["returns"] == t["returns"]:
            solutions.append(s.copy())
        results = DFS_Search(t, g, r, s)
        solutions.extend(results)
        r.remove(tool["returns"])
        s.remove(tool)
    return solutions                                    ▷ Return
end function
```

Greedy Strategy. This strategy selects the tool node with the highest score at each step, where the score indicates the relevance of the tool to the task. Greedy search is fast and simple, but it may not find the optimal solution or even any solution at all.

Beam Strategy. It only keeps the k best tools according to their assessment scores. Beam search can expand the search space but reduce the search efficiency slightly.

Adaptive Strategy. This is a variant of beam search where it dynamically adjusts the beam size by choosing the tools with scores higher than a fixed threshold, which is a trade-off between exploration and exploitation.

Exhaustive Strategy. This strategy explores all possible paths from the start node to the terminal node. The exhaustive search is guaranteed to find an optimal solution if one exists, but it may be very slow during the search.

The impacts of different search strategies are studied in Sec. 4.4. The search process, akin to a brainstorm or mind map, hunts for potential solutions.

Solutions Post-processing After ToG searches the solutions, we design *solution expert* and *resource expert* to post-process solutions, which both employ prompt engineering to instruct the language model $\mathcal{M}$ to complete the tasks.

Specifically, *solution expert* to select the optimal one among all solution candidates and *resource expert* to infer the remaining arguments for tools, respectively.

3.3 Solution Execution

Once the task solutions are completed, they are passed to the *execution engine* to obtain results, as shown in Fig. 2. In this stage, the execution engine initially parses the solutions into a sequence of *Actions*. Each action is associated with particular tool services, which could be implemented via either handcrafted mapping tables or an automatic scheduler based on some strategies. Our execution engine empowers the system with the flexibility to schedule diverse tools based on users' preferences.

The parsed actions are automatically executed by scheduling the action to the local, remote, or hybrid endpoints. Multiple independent subtasks would be executed in parallel to improve efficiency. Besides, we maintain a state memory storing all the intermediate results, including their values and types. This enables the running-time automatic correction for the action parameters.

Response Generation. The unprocessed results may lack comprehensiveness and clarity, potentially making it difficult for users to understand. Therefore, we introduce a module to aggregate all the execution results and generate user-friendly responses. This is achieved by prompting the LLMs, such as ChatGPT, with the user request, action list, and execution results and asking them to summarize the answers intelligently.

3.4 The Choices of Language Model

To systematically verify the performance of the overall framework, we here discuss different options for the language model. One feasible yet direct choice is to use off-the-shelf **large language models** (LLMs) such as ChatGPT or Llama 2 [41], which are pre-trained on large-scale text corpora and can handle various NLP tasks. These LLMs are readily available. We design a series of elaborate prompts for task decomposition, tool assessment and solutions post-processing. We call this variant as ControlLLM-ChatGPT. In this way, we avoid training a language model from scratch. However, they may lead to low performance as they are not trained for our requirements. The alternative choice of $\mathcal{M}$, termed as ControlLLM-LLaMA, is to finetune a language model (*e.g.*, LLaMA) by using self-instruct method [44]. The advantage of this variant is that it can achieve high performance by adapting to the data and the task. Nevertheless, it requires lots of GPUs to train the model and may suffer from overfitting issues.

All prompts used to instruct LLMs are shown in the Appendix for clarity.

4 Experiments

4.1 Benchmark

We build a benchmark that is used to evaluate our proposed framework compared with other state-of-the-art methods. This benchmark consists of a set of

tasks that require various tools to solve complex problems collaboratively. It is designed to cover different task domains, such as question answering, image generation, image editing, image perception, visual question answering, *etc*. Despite the fact that our method supports more tools than other systems, in order to make fair comparisons, we only evaluate and test on the intersection of toolsets from different methods [24, 36, 47, 49], all of which share comparable toolsets.

This benchmark includes about 100 instructions, which are classified into three levels of difficulty: easy (< 2 APIs), medium (2 or 3 APIs), and hard(> 3 APIs). We use test instructions with various levels to meticulously validate the ability of different methods. However, this manner may cause a critical problem: whether the scale of this benchmark is enough for evaluation. Since some methods [24, 47] do not support formatted outputs for their task planning, it is difficult to automatically evaluate their capabilities with a large batch of instructions. In addition, we observe the output format and toolset between different methods [11, 36, 38, 49] is also extremely inconsistent with each other. It may lead to biased and unfair comparisons if using LLMs or rule-based methods to evaluate them. So, in this work, we choose to evaluate their performance via a multi-person voting approach. As a result, we have to limit the scale of our benchmark. For this concern, we add extra experiments on a large-scale test set to verify the efficacy of our method. Generally, we believe that our experiments are able to provide a comprehensive comparison of the tool control capabilities of different methods. In the appendix, we present some instruction samples from our benchmark. It is noticeable that there is no absolute relationship between difficulty and length of instruction.

4.2 Evaluation Protocol

Effectively evaluating the performance of tool-augmented LLMs remains a challenging task. The challenges stems from several factors, including the inherent ambiguities in natural language, the absence of shared benchmarks, and formatted solutions for systematically assessing different methods.

In addition, we find the APIs of tools in different methods are slightly inconsistent. It is hard to annotate all feasible solutions for each method. As such, we adopt an evaluation protocol via a multi-person voting approach with three experts. The protocol breaks down the evaluation into three main aspects: tool selection, argument assignment, and overall solution evaluation. Please note that the evaluation protocol is independent of the tools' capabilities. When the tools and their input arguments are correct, we do not account for the case where the output fails to satisfy the user's expectations due to the limitations of tools.

Metrics for Tool Selection: A) Irrelevant Tool Inclusion Rate (*abbr.* IR): This metric gauges the performance of the method in excluding irrelevant tools. It measures the proportion of the predicted solutions that contain the irrelevant tools. A higher IR indicates that the method tends to include more unnecessary tools, potentially hindering effective task planning; B) Necessary Tool Inclusion Rate (*abbr.* NR): This metric assesses the inclusion of necessary tools in the predicted solution but without considering whether the arguments of tools are

Table 2: Comparisons with the state-of-the-art methods. ↓ means the smaller the better, ↑ means the larger the better. The results of these methods [24, 36, 47, 49] are reproduced on our own benchmark. ∗ denotes the default setting of ControlLLM if not stated.

Methods	Tool		Argument		Solution Evaluation ↑			
	IR ↓	NR ↑	HR ↓	CR ↑	All	Easy	Medium	Hard
HuggingGPT [36]	0.45	0.64	0.16	0.69	0.59	0.73	0.50	0.33
Visual ChatGPT [47]	0.26	0.58	0.09	0.76	0.57	0.73	0.63	0.10
InternGPT [24]	0.12	0.51	0.49	0.43	0.44	0.60	0.46	0.00
VISPROG [11]	0.50	0.77	0.06	0.84	0.59	0.71	0.58	0.29
ViperGPT [38]	0.13	0.86	0.07	0.93	0.75	0.76	0.92	0.52
GPT4Tools [49]	0.19	0.44	0.28	0.72	0.43	0.64	0.33	0.00
ControlLLM-ChatGPT	0.16	0.63	0.83	0.83	0.64	0.71	0.67	0.43
ControlLLM-LLaMA	0.06	**0.95**	0.02	0.98	0.91	0.98	0.88	0.76
ControlLLM-Mix*	**0.03**	0.93	**0.02**	**0.98**	**0.93**	**0.98**	**0.96**	**0.81**

correct. If NR is high, it indicates the method has strong capabilities in tool selection.

Metrics for Argument Assignment: A) Resource Hallucination Rate (*abbr.* HR): This indicator reveals the extent of hallucination in the method's responses when inferring the arguments for tools. It measures whether all arguments of the tools used in the predicted solution exist. A lower HR suggests that the method is less prone to generating hallucinated content. B) Resource Type Consistency Rate (*abbr.* CR): This metric examines whether the types of input resources in the predicted solution match those of the corresponding tools. It evaluates the method's ability to ensure consistency of input types of tools.

Solution Evaluation (SE) measures the success rate of all generated solutions on our benchmark. It only considers whether the output solution can effectively address the user's problem, irrespective of whether it contains irrelevant tools. A higher score in the solution evaluation indicates a stronger capability for task planning.

4.3 Quantitative Comparisons

In this section, we give a comprehensive analysis of ControlLLM to compare with state-of-the-art methods, as summarized in Table 2. We provide three implementations in supplementary materials for our method: a) ControlLLM-ChatGPT leverages the ChatGPT-3.5 as language model $\mathcal{M}$; b) ControlLLM-LLaMA that finetunes a LLaMA-7B as a language model $\mathcal{M}$; c) ControlLLM-Mix is regarded as our default setting, which finetunes LLaMA-7B as a task decomposer in the first stage while the remaining modules employ the ChatGPT to finish the tasks. We find that instruction-tuned LLaMA is good at task decomposition, while

Table 3: The evaluation for different search strategies. We count the average number of visited tools to denote the time complexities.

Search Strategies	Tool		Argument		Solution Evaluation ↑				Time Complexities
	IR ↓	NR ↑	HR ↓	CR ↑	All	Easy	Meduim	Hard	
Greedy	0.19	0.49	0.24	0.76	0.49	0.56	0.58	0.19	4.07
Beam ($k = 3$)	0.14	0.88	0.01	0.99	0.88	0.96	0.79	0.76	121.29
Adaptive	**0.03**	0.93	0.02	0.98	0.93	0.98	0.96	0.81	236.49
Exhaustive	0.06	**0.97**	**0.01**	**0.99**	**0.97**	**1.00**	**0.96**	**0.91**	3444.23

Table 4: The effects of task decomposition with regard to different LLMs. *PK* denotes "prior knowledge". We find, if adding prior knowledge, such as which tools might be used, into the subtask description, the performance of task planning can be evidently improved.

Task Decomp.	LLMs	Tool		Argument		Solution Evaluation ↑			
		IR ↓	NR ↑	HR ↓	CR ↑	All	Easy	Meduim	Hard
w/o PK	Llama2-13B	0.28	0.71	0.01	0.99	0.68	0.87	0.50	0.38
	ChatGPT-3.5	0.13	0.84	0.01	0.99	0.83	0.99	0.67	0.57
	GPT-4	0.06	0.91	0.03	0.97	0.91	0.98	0.83	0.81
w/ PK	Llama2-13B	0.12	0.83	0.04	0.95	0.82	0.95	0.71	0.62
	ChatGPT-3.5	0.03	0.93	0.02	0.98	0.93	0.98	0.96	0.81
	GPT-4	**0.01**	**0.98**	**0.02**	**0.98**	**0.98**	**1.00**	**1.00**	**0.91**

ChatGPT has a strong ability to select tools. ControlLLM-Mix (*abbr.* ControlLLM) combines the advantages of the other two variants. It thus achieves the most promising performance on our benchmark. In addition, some methods [24,47] built upon Chain of Thought are also selected to validate the efficacy of proposed ToG.

Specifically, ControlLLM excels in several key aspects. Notably, it achieves the lowest Irrelevant Tool Inclusion Rate (IR) as well as the highest Necessary Tool Inclusion Rate, indicating its ability in effective yet efficient task planning. Furthermore, ControlLLM demonstrates superior performance in argument assignment, with the lowest Argument Hallucination Rate (HR) of 0.02 and the highest Argument Type Consistency Rate (CR) of 0.98. These results underscore its ability to generate accurate and consistent arguments, addressing a challenge in tool-augmented LLMs. In the solution evaluation, ControlLLM maintains its lead with a score of 0.93, indicating its effectiveness in resolving user requests. When compared with HuggingGPT [36] and Visual ChatGPT [47] that both rely on ChatGPT to perform task planning, our ControlLLM-ChatGPT still outperform them in the solution evaluation. This proves the efficacy of our ToG. In summary, ControlLLM exhibits remarkable performance in all proposed metrics when compared with state-of-the-art methods in this field.

4.4 Ablation Studies

Table 3 investigates the impact of different search strategies within our Thoughts-on-Graph. We observe that the exhaustive search strategy outperforms others in most metrics, but this strategy is not so practical due to its time-consuming search. On the other hand, the greedy search strategy achieves the lowest performance. It thus usually fails to find the solution, especially in complicated cases, because the greedy search strategy can not search for a feasible path based on the tool with a high score when the tool assessment is not so accurate. As the greedy strategy selects the best-matched tool assessed by LLMs and Tree of Thoughts [50] using DFS explores the most promising state first, they are equivalent to some extent. In addition, the adaptive strategy strikes a balance between performance metrics and time complexities, offering competitive results in most aspects. To trade-off between time and accuracy, we thus choose the adaptive strategy as our default setting.

In Table 4, we conduct ablation studies to evaluate the impact of language models on ControlLLM-Mix. We find language models play a decisive role in tool selection and solution ranking. The more powerful the language model, the higher the score of solution evaluation. Furthermore, we investigate the effects of incorporating prior knowledge into the subtask descriptions in the task decomposition. The method without prior knowledge usually directly uses the user's request as a subtask description and does not offer any hints or suggestions on tool selections in the subtask description. In contrast, in the variant with prior knowledge, we add prior knowledge to the subtask description. The prior knowledge indeed improves the necessary tool inclusion rate (NR) and reduces the chance of selecting irrelevant tools (IR) when using the same language model. It provides the inspiration to further improve the overall system.

More experiments, including user study, can be found in the appendix.

4.5 Qualitative Analyses

Fig. 3 shows two simple cases to illustrate the capabilities of our ControlLLM in task planning. In contrast to HuggingGPT [36], our method is able to generate more diverse solutions to meet users' expectations, thanks to the Thoughts-on-Graph paradigm. In the appendix, we also provide extensive case studies across different modalities to validate the user experience for our method in practice.

5 Conclusion

In this paper, we propose **ControlLLM**, a multi-modal interaction framework that can accurately control tool usage across various domains, including text, image, audio, video, *etc*. The proposed framework consists of three key stages: (1) *task decomposition* to concrete the objective of the task, (2) a *Thoughts-on-Graph* (ToG) paradigm to search the optimal solution path on the constructed tool graph, and (3) an *execution engine* with a versatile toolbox to execute

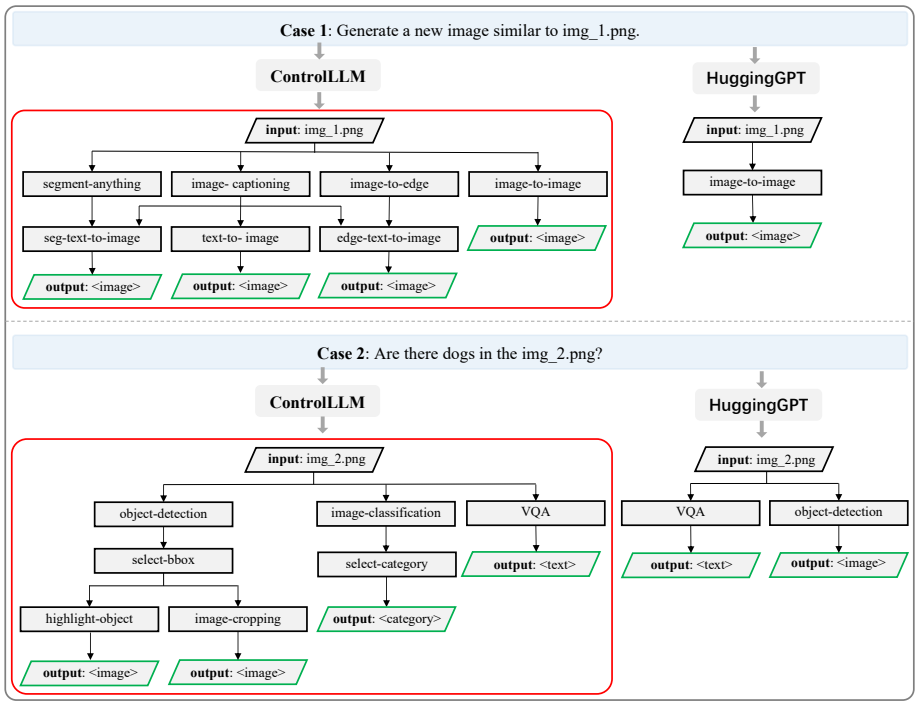

Fig. 3: Qualitative comparisons of task planning. We here use two simple cases to illustrate the differences between two different methods of task planning. Here, each output node is generated by different solution paths. More demo visualizations can be referred to in the appendix.

solution efficiently. We conduct extensive experiments and demonstrate that our ControlLLM achieves superior performance regarding tool selection, argument assignment, and overall solution effectiveness compared to existing methods.

Nevertheless, **we acknowledge that this work still has some limitations**. Since the goal of this work is to improve the accuracy of tool usage, even if the solution is theoretically feasible, we cannot guarantee that the output from tools is always correct. On the other hand, due to the inherent ambiguity in natural language, it is difficult to ensure that the optimal solution selected is consistent with the user's goal. In this case, we can only provide more alternative solutions searched by ToG for users to choose from if the optimal solution fails. In addition, as ToG is based on the depth-first search algorithm to find the solution, it is time-consuming when performing task planning, but this disadvantage can be optimized with some engineering tricks. We also expect to see some works that can improve the runtime efficiency of ToG in the future.

Acknowledgements

The work is supported by the National Key R&D Program of China (No. 2022ZD0161300), the National Natural Science Foundation of China (No. 62376134, 62372223), and Guangdong Science and Technology Department (Grant No. 2023A0505010004).

References

1. Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al.: Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691 (2022)
2. Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Gianinazzi, L., Gajda, J., Lehmann, T., Podstawski, M., Niewiadomski, H., Nyczyk, P., Hoefler, T.: Graph of Thoughts: Solving Elaborate Problems with Large Language Models (2023)
3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
4. Chiang, W.L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J.E., Stoica, I., Xing, E.P.: Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality (March 2023), `https://lmsys.org/blog/2023-03-30-vicuna/`
5. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311 (2022)
6. Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al.: Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416 (2022)
7. Dai, W., Li, J., Li, D., Tiong, A.M.H., Zhao, J., Wang, W., Li, B., Fung, P., Hoi, S.: Instructblip: Towards general-purpose vision-language models with instruction tuning (2023)
8. Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., Sui, Z.: A survey for in-context learning. arXiv preprint arXiv:2301.00234 (2022)
9. Dong, R., Han, C., Peng, Y., Qi, Z., Ge, Z., Yang, J., Zhao, L., Sun, J., Zhou, H., Wei, H., et al.: Dreamllm: Synergistic multimodal comprehension and creation. arXiv preprint arXiv:2309.11499 (2023)
10. Gao, Z., Du, Y., Zhang, X., Ma, X., Han, W., Zhu, S.C., Li, Q.: Clova: A closed-loop visual assistant with tool usage and update. arXiv preprint arXiv:2312.10908 (2023)
11. Gupta, T., Kembhavi, A.: Visual programming: Compositional visual reasoning without training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14953–14962 (2023)
12. Hao, S., Liu, T., Wang, Z., Hu, Z.: Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. arXiv preprint arXiv:2305.11554 (2023)
13. He, Y., Liu, Z., Chen, J., Tian, Z., Liu, H., Chi, X., Liu, R., Yuan, R., Xing, Y., Wang, W., et al.: Llms meet multimodal generation and editing: A survey. arXiv preprint arXiv:2405.19334 (2024)
14. Ho, N., Schmid, L., Yun, S.Y.: Large language models are reasoning teachers. arXiv preprint arXiv:2212.10071 (2022)

15. Jiang, Y., Yan, X., Ji, G.P., Fu, K., Sun, M., Xiong, H., Fan, D.P., Khan, F.S.: Effectiveness assessment of recent large vision-language models. arXiv preprint arXiv:2403.04306 (2024)
16. Koh, J.Y., Fried, D., Salakhutdinov, R.: Generating images with multimodal language models. arXiv preprint arXiv:2305.17216 (2023)
17. Lai, X., Tian, Z., Chen, Y., Li, Y., Yuan, Y., Liu, S., Jia, J.: Lisa: Reasoning segmentation via large language model. arXiv preprint arXiv:2308.00692 (2023)
18. Latombe, J.C.: Robot motion planning, vol. 124. Springer Science & Business Media (2012)
19. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv preprint arXiv:2301.12597 (2023)
20. Li, J., Li, D., Xiong, C., Hoi, S.: Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: International Conference on Machine Learning. pp. 12888–12900. PMLR (2022)
21. Liu, C., Jiang, X., Ding, H.: Primitivenet: decomposing the global constraints for referring segmentation. Visual Intelligence **2**(1), 16 (2024)
22. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. arXiv preprint arXiv:2304.08485 (2023)
23. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: NeurIPS (2023)
24. Liu, Z., He, Y., Wang, W., Wang, W., Wang, Y., Chen, S., Zhang, Q., Lai, Z., Yang, Y., Li, Q., Yu, J., et al.: Interngpt: Solving vision-centric tasks by interacting with chatbots beyond language. arXiv preprint arXiv:2305.05662 (2023)
25. Ma, L., Han, J., Wang, Z., Zhang, D.: Cephgpt-4: An interactive multimodal cephalometric measurement and diagnostic system with visual large language model. arXiv preprint arXiv:2307.07518 (2023)
26. Moon, S., Madotto, A., Lin, Z., Nagarajan, T., Smith, M., Jain, S., Yeh, C.F., Murugesan, P., Heidari, P., Liu, Y., et al.: Anymal: An efficient and scalable any-modality augmented language model. arXiv preprint arXiv:2309.16058 (2023)
27. Mu, Y., Zhang, Q., Hu, M., Wang, W., Ding, M., Jin, J., Wang, B., Dai, J., Qiao, Y., Luo, P.: Embodiedgpt: Vision-language pre-training via embodied chain of thought. arXiv preprint arXiv:2305.15021 (2023)
28. OpenAI: Chatgpt (Mar 14 version) [large language model]. 6 (2023)
29. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems **35**, 27730–27744 (2022)
30. Parisi, A., Zhao, Y., Fiedel, N.: Talm: Tool augmented language models. arXiv preprint arXiv:2205.12255 (2022)
31. Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: Large language model connected with massive apis. arXiv preprint arXiv:2305.15334 (2023)
32. Peng, Z., Wang, W., Dong, L., Hao, Y., Huang, S., Ma, S., Wei, F.: Kosmos-2: Grounding multimodal large language models to the world. arXiv preprint arXiv:2306.14824 (2023)
33. Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., et al.: Toolllm: Facilitating large language models to master 16000+ real-world apis. arXiv preprint arXiv:2307.16789 (2023)
34. Rae, J.W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al.: Scaling language models: Methods, analysis & insights from training gopher. arXiv preprint arXiv:2112.11446 (2021)

35. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language models can teach themselves to use tools. arXiv preprint arXiv:2302.04761 (2023)
36. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. arXiv preprint arXiv:2303.17580 (2023)
37. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. nature **550**(7676), 354–359 (2017)
38. Surís, D., Menon, S., Vondrick, C.: Vipergpt: Visual inference via python execution for reasoning. Proceedings of IEEE International Conference on Computer Vision (ICCV) (2023)
39. Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., Sun, L.: Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. arXiv preprint arXiv:2306.05301 (2023)
40. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
41. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
42. Vemprala, S., Bonatti, R., Bucker, A., Kapoor, A.: Chatgpt for robotics: Design principles and model abilities. Microsoft Auton. Syst. Robot. Res **2**, 20 (2023)
43. Wang, W., Chen, Z., Chen, X., Wu, J., Zhu, X., Zeng, G., Luo, P., Lu, T., Zhou, J., Qiao, Y., et al.: Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. arXiv preprint arXiv:2305.11175 (2023)
44. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
45. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)
46. Weiss, S.M., Kulikowski, C.A., Amarel, S., Safir, A.: A model-based method for computer-aided medical decision-making. Artificial intelligence **11**(1-2), 145–172 (1978)
47. Wu, C., Yin, S., Qi, W., Wang, X., Tang, Z., Duan, N.: Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671 (2023)
48. Wu, S., Fei, H., Qu, L., Ji, W., Chua, T.S.: Next-gpt: Any-to-any multimodal llm. arXiv preprint arXiv:2309.05519 (2023)
49. Yang, R., Song, L., Li, Y., Zhao, S., Ge, Y., Li, X., Shan, Y.: Gpt4tools: Teaching large language model to use tools via self-instruction (2023)
50. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T.L., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. arXiv preprint arXiv:2305.10601 (2023)
51. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629 (2022)
52. Yao, Y., Li, Z., Zhao, H.: Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. arXiv preprint arXiv:2305.16582 (2023)

53. You, H., Zhang, H., Gan, Z., Du, X., Zhang, B., Wang, Z., Cao, L., Chang, S.F., Yang, Y.: Ferret: Refer and ground anything anywhere at any granularity (2023)
54. Zeng, A., Liu, X., Du, Z., Wang, Z., Lai, H., Ding, M., Yang, Z., Xu, Y., Zheng, W., Xia, X., et al.: Glm-130b: An open bilingual pre-trained model. arXiv preprint arXiv:2210.02414 (2022)
55. Zhang, R., Han, J., Zhou, A., Hu, X., Yan, S., Lu, P., Li, H., Gao, P., Qiao, Y.: Llama-adapter: Efficient fine-tuning of language models with zero-init attention. arXiv preprint arXiv:2303.16199 (2023)
56. Zhang, S., Sun, P., Chen, S., Xiao, M., Shao, W., Zhang, W., Chen, K., Luo, P.: Gpt4roi: Instruction tuning large language model on region-of-interest. arXiv preprint arXiv:2307.03601 (2023)
57. Zheng, K., He, X., Wang, X.E.: Minigpt-5: Interleaved vision-and-language generation via generative vokens (2023)
58. Zheng, K., He, X., Wang, X.E.: Minigpt-5: Interleaved vision-and-language generation via generative vokens. arXiv preprint arXiv:2310.02239 (2023)
59. Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al.: Least-to-most prompting enables complex reasoning in large language models. arXiv preprint arXiv:2205.10625 (2022)