

Research article

LLM-Collab: a framework for enhancing task planning via chain-of-thought and multi-agent collaboration

Hong Cao¹, Rong Ma², Yanlong Zhai^{1,*} and Jun Shen³

¹ School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China

² Science and Technology on Space System Simulation Laboratory, Beijing Simulation Center, Beijing, China

³ School of Computing and Information Technology, University of Wollongong, Wollongong, Australia

* **Correspondence:** Email: ylzhai@bit.edu.cn; Tel: +861068913536.

Academic Editor: Jidong Yang

Abstract: Large language models have shown strong capabilities in performing natural language planning tasks, largely due to the chain-of-thought method, which enhances their ability to solve complex tasks through explicit intermediate inference. However, they face challenges in acquiring new knowledge, executing calculations, and interacting with the environment. Although previous work has enabled large language models to use external tools to improve reasoning and environmental interaction, there was no scalable or cohesive structure for these technologies. In this paper, we present LLM-Collab, where Collab represents the cooperative interaction between two AI agents, and the large language model plays a key role in the creation of AI agents. For this method, we took large language models as the reasoning core for AI agents and designed two AI agents to cooperate on the planning tasks: One as an analyst for tool selection and phase validation, and the other as an executor of specific tasks. Our method provided a comprehensive list of external tools to facilitate the invocation and integration of agents, ensuring a seamless collaboration process. This paradigm established a unified framework for autonomous task-solving based on massive language models by demonstrating how language communication and tool selection enable multi-agent collaboration.

Keywords: chain-of-thought; task planning; large language models; AI agent; LLM-Collab

1. Introduction

In the progression of artificial intelligence (AI), machine learning has been crucial in enhancing the comprehension and processing of natural language. The emergence of large language models (LLMs)

marks a significant milestone in AI, demonstrating exceptional proficiency in language understanding and task planning. The powerful role-assumption capabilities of LLMs have greatly reduced the requirement for custom model designs, increasing their efficacy and adaptability [1]. This versatility is further amplified by the integration of autonomous agents [2], which enhance model competencies in domains such as task decomposition, contextual learning [3], and the selection of external tools [4]. These studies provide possibilities for the application of LLMs in task planning.

Traditional approaches in the field of task planning rely predominantly on symbolic methods or reinforcement learning-based techniques, such as the Planning Domain Definition Language (PDDL) [5] or policy learning [6]. However, these conventional methodologies face several inherent limitations. In particular, symbolic approaches require the conversion of adaptable, naturally expressed problems into a symbolic model, which frequently requires the knowledge and arduous work of human experts. This conversion step can be time-consuming and prone to errors, highlighting the need for more adaptive and efficient planning strategies. Reinforcement learning algorithms typically require a substantial volume of samples to learn effective strategies. However, if data gathering is time-consuming or costly, this need may make the strategy unfeasible or prohibitively expensive.

The integration of LLMs as the central component of autonomous agents for task planning has marked significant advances in recent research efforts [7]. The complexity inherent in task planning requires the deconstruction of tasks and the discerning use of tools meticulously customized to address the distinctive demands of each task. This procedure necessitates a thorough comprehension and analysis made possible by language models, which also include the verification and improvement of the results obtained from outside resources. To address these complexities, researchers are diligently investigating methodologies for integrating the semantic comprehension of large language models with the practical imperatives of task planning. For example, integrating tools such as conventional planners [8], procedural auxiliary language models (PAL) [9], and search algorithms [10] has substantially improved the efficacy of models in inference tasks. The LLM+P [8] framework uses PDDL as a foundational tool to articulate and structure the planning process. PDDL plays a key role in decomposing complex issues into smaller, more manageable parts that can be handled by certain planning models. This approach facilitates the generation of a plan that is subsequently articulated in natural language. However, the current implementation of LLM+P is limited to a single interaction or dialogue, which does not allow for a comprehensive analysis or iterative refinement of the planning outcomes.

Disparities between these technologies have led to disjoint methodologies in various phases, requiring bespoke designs for tasks in divergent scenarios. This has resulted in a fragmented developmental pathway within the field. Motivated by the unexploited potential of autonomous agents, our goal is to develop an integrated framework for agent-based task planning. We advocate for the employment of two LLM-driven agents to enable intelligent planning within multitask environments employing the chain of thought (cot) [11] to deconstruct the planning process and promote collaboration in different phases.

In this paper, we introduce a methodology called LLM-Collab. Our framework improves model performance in complex inference tasks and provides a scalable solution to complex task planning challenges. Our method outlines task planning in four key phases: Task analysis, task planning, program validation and improvement, and result consolidation. Each phase is further decomposed into a series of subtasks through a chain of thought methodology, facilitating iterative planning

through multi-turn dialogues. The framework includes a centralized list of external tools for agent use and supports future expansion (see Figure 1 for an example). Furthermore, we advocate for the development of specialized models to improve planning precision, enhancing the flexibility and professionalism of the framework through collaboration between LLMs and smaller models [12]. Our findings show that combining LLMs with intelligent agents leads to more automated and effective task planning.

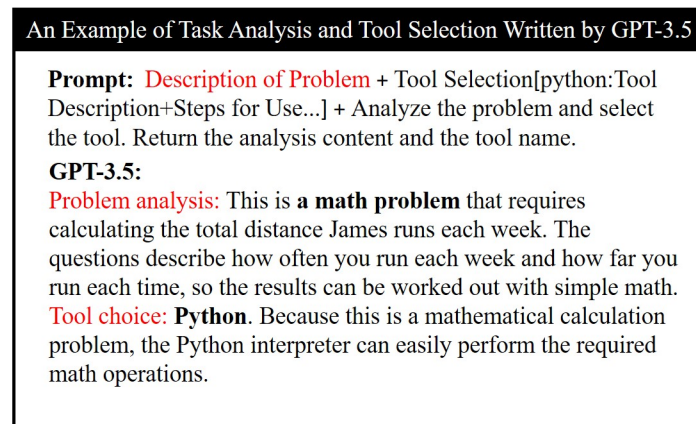


Figure 1. Guide the LLMs to analyze the problem and provide them with a list of external tools and steps to use. In the planning phase, LLM-Collab makes planning calls according to the usage steps.

2. Enhancing task planning with LLMs

In this section, we commence with a foundational exploration of the cot methodologies. Following this, we elucidate how the LLM confers sophisticated language processing capabilities upon intelligent agents, thereby facilitating their interaction with the environment and enhancing the efficiency with which they execute tasks. The discourse culminates with an exploration of scholarly efforts aimed at augmenting the capabilities of LLMs through the integration of supplementary external modules.

2.1. Chain of thought

The *chain of thought* is an optimized prompt strategy that aims to improve the comprehension of internal decision-making processes within large language models. This enhancement is achieved by providing a clearer and more detailed explanation of the reasoning process, articulated through a sequence of logical intermediate steps.

The cot methodology is a critical strategy for increasing the inferential capabilities of LLMs. Cot enables a notable improvement in handling intricate inferential tasks that need multi-step reasoning by systematically encouraging the model to articulate intermediate reasoning steps. This approach effectively disassembles intricate problems into a sequence of more manageable sub-problems. Furthermore, it mirrors human cognitive processes, thereby bolstering the interpretability and reliability of the model's decision-making processes. Incorporation of cot has shown improvements in accuracy and efficiency in various domains, including arithmetic, common sense reasoning [13], and symbolic reasoning [14]. Even a modest number of cot examples has been shown to substantially

elevate the performance of LLMs on pertinent benchmarks. An illustrative example of a cot prompt is depicted in Figure 2.

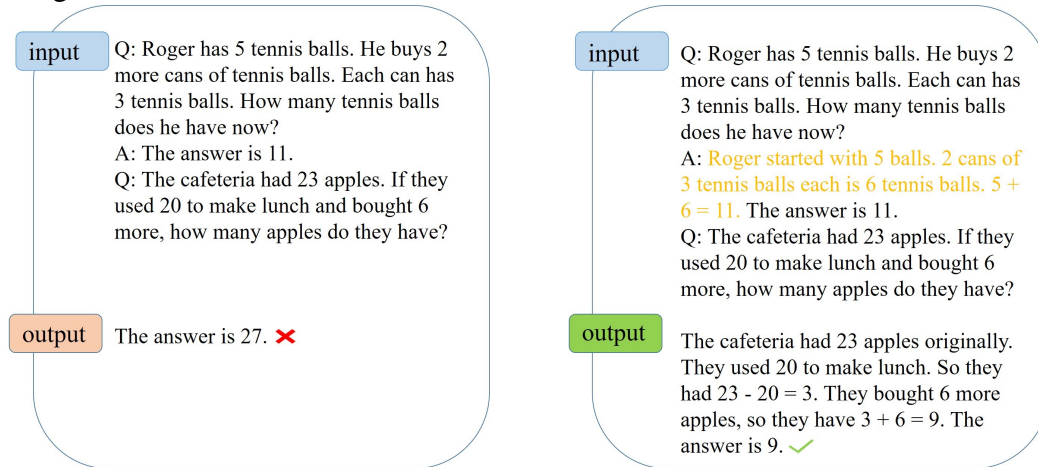


Figure 2. The chain of thought prompting technique empowers LLMs to adeptly handle intricate arithmetic, common sense, and symbolic reasoning challenges. This method illuminates the step-by-step reasoning processes, emphasizing how it significantly enhances the depth and breadth of a model’s problem-solving capabilities.

The cot approach is characterized by its efficacy in simplifying complex issues through a stepwise breakdown, optimizing the allocation of computational resources, and enhancing the interpretability of the model’s inferential procedures. This clarity facilitates a deeper comprehension of the model’s reasoning trajectory and enables the identification and correction of logical inconsistencies. In particular, the efficacy of cot is positively correlated with the scale of the model, with larger models demonstrating superior performance in constructing coherent thought sequences. This correlation underscores the importance of model size as a determinant in realizing the full reasoning capacity of LLMs. The integration of the cot with advanced task planning methodologies has produced a framework capable of effectively addressing real-world planning challenges.

The versatility of the cot method is evident in its applicability in diverse domains. It is adept at addressing not only mathematical problems but also tasks that require common sense reasoning, encompassing scenarios involving physical phenomena and human interactions. This capability is predicated on the model’s comprehension of extensive background knowledge, thereby underscoring the potential of cot to expand the spectrum of tasks that LLMs can adeptly execute. As the cot methodology continues to advance, it holds promise for the future development of sophisticated AI systems poised to excel in intricate planning and reasoning endeavors.

Within the LLM-Collab framework, the cot methodology is deployed to augment the inferential prowess of an individual model and to foster collaborative interactions between distinct roles: the analyst and the executor. The analyst assumes the role of overseeing strategic planning and macro-level decision making, whereas the executor is tasked with the implementation of specific operational directives.

2.2. Agent

In artificial intelligence, LLMs extend beyond text generation alone. They serve as the cognitive core of AI agents, endowed with the ability to make and plan autonomous decisions [15]. The

integration of these models has transformed AI agents into complex entities equipped with capabilities for environmental perception, information processing, and the execution of actions to achieve specified objectives. The agents' ability to interact with their environment has been greatly enhanced by this combination, providing a strong basis for reasoning, memory access, and knowledge generalization.

As LLM-based agents have matured, they have been deployed in various scenarios, including single agent tasks, complex multi-agent systems, and human-agent collaborations [16].

In single-agent deployments, these agents have demonstrated efficiency in performing everyday tasks such as web navigation and household chores by breaking down high-level instructions into manageable subtasks. Their proficiency in understanding and generation of natural languages has been instrumental in facilitating human-agent interactions, enabling them to act as assistants in roles such as education, healthcare, and customer service. TheAgentCompany [17] conducted a comprehensive benchmarking study on the performance of LLM agents in practical applications, thereby contributing an evaluation framework specifically targeting single-agent systems. ReAct [10] integrates reasoning and action in a single agent model, emphasizing the importance of structured task decomposition and planning to enable complex problem solving in LLM. GeoAgent [18] is an agent based on a large language model that incorporates a code interpreter, static analysis, and retrieval-augmented generation to facilitate automated analysis of geospatial datasets.

In the context of multi-agent systems, the collaborative and competitive interactions among agents have opened new avenues for collective intelligence. Through cooperative efforts, agents can complement each other's skills and knowledge, leading to more efficient and effective outcomes. However, adversarial interactions have led agents to refine their strategies and improve their performance, similar to a process of debate and argumentation. For example, ChatDev [19] is an innovative software development framework that leverages specialized LLM-driven agents, guiding them through chat chains on how to communicate and collaborate during the design, coding, and testing phases. These agents generate solutions through multiple rounds of dialogue, using natural language for system design and programming languages for debugging, thus employing language as a unified bridge for autonomous task solving. ChatDev subdivides each phase into smaller subtasks via chat chains and reduces coding hallucinations through communication de-hallucination mechanisms, thus facilitating the study of multi-agent collaboration. LongAgent [20] enhances the contextual capacity of the language model to 128k through the synergistic efforts of multi-agent collaboration, thereby illustrating the potential of multi-agent systems to augment the model's capabilities.

In the context of human-agent collaborations, the synergy between humans and intelligent agents is critical for accomplishing specific tasks and solving problems collectively. This collaborative model integrates human creativity and judgment with the data processing capabilities and real-time responsiveness of intelligent agents, resulting in a more efficient and intelligent operational approach. In practical applications, human-machine collaboration has a broad spectrum of utility across domains. CoSearchAgent [21] is a lightweight collaborative search agent that leverages large language models to augment human search capabilities. ReHAC [22] is a human-agent collaboration approach based on reinforcement learning that enhances the performance of agents based on LLM in complex tasks. Some studies have investigated the impact of varying levels of artificial intelligence scaffolding on human writing quality and productivity [23], from none to high, and have found that high scaffolding significantly improves both, especially benefiting non-regular writers and less tech-savvy users.

2.3. External tool

In the contemporary landscape of AI, the complexity of task planning demands sophisticated systems with advanced natural language processing capabilities and the ability to effectively leverage external resources. The integration of external toolsets serves as a catalytic agent, significantly enhancing the capabilities of LLMs in task planning.

Researchers have actively investigated methods to enhance the reasoning capabilities of LLMs through the integration of external tools. The integration of Planning Domain Definition Language [24, 25] planners substantially enhances models by providing a structured methodology to understand and resolve task planning challenges. The formal linguistic framework of PDDL enables LLMs to articulate planning problems methodically, which is essential to automate complex planning processes. Researchers are also trying to complement and extend the capabilities of LLMs with sophisticated external tools, including application programming interfaces, databases, and file management systems.

The integration of tools from various domains is essential for fostering interdisciplinary approaches to problem solving. When equipped with a suite of tools that cater to multiple domains, LLMs are enabled to address sophisticated tasks that require a combination of linguistic interpretation and visual analysis. This multifaceted capability enables them to effectively manage a broader spectrum of applications, enhancing their versatility in handling complex multimodal challenges.

Ultimately, the ensemble of external tools facilitates intrinsic validation and error rectification mechanisms throughout the task execution phase. This capability ensures that the solutions generated are both logically sound and applicable in real-world contexts. The capacity for autonomous error correction not only bolsters the dependability of the task execution process, but also promotes ongoing learning and enhancement within the model framework.

3. LLM-Collab

In this section, we describe our method. *LLM-Collab* is an innovative intelligent collaboration framework that integrates intelligent agents with different professional roles in the core phases of task planning, as shown in Figure 3.

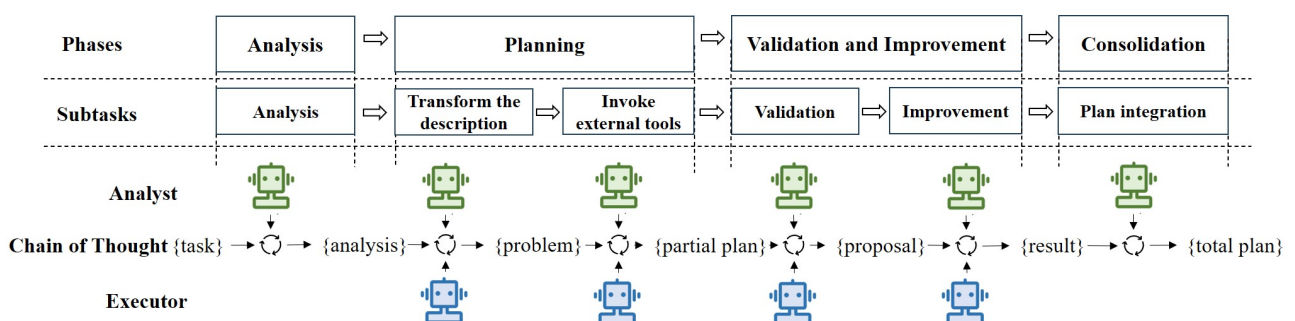


Figure 3. After receiving the initial task request, the two agents participate in multiple rounds of communication and execute the instructions according to the workflow of the chain structure. Collaboratively and autonomously perform a series of sub-tasks to form a comprehensive planning scenario.

3.1. Task planning

The task planning phase is a pivotal step in implementing complex task solutions. This phase involves specific processes:

- **Analysis:** Initially, the analyst is required to understand the planning inquiries presented by the user and to elucidate the objectives and requirements of the task [26]. This demands an insightful understanding of the user's intent and a thorough grasp of the task context. In this phase, we integrate the problem into a pre-defined chain of thought template, providing the model with a selection of available tools and their descriptions. This guides the model in analyzing the problem and making an informed choice of tools to employ. As shown in Figure 1.
- **Transform the description:** In the LLM-Collab framework, based on the characteristics and requirements of the problem, appropriate tools are selected from external tools to support the execution of subsequent tasks. We describe the functions and applicable scenarios of the available tools in the form of a list to improve the possibility of correct invocation of the model. The user questions are then carefully translated into detailed task descriptions so that the appropriate tools can be entered accurately to ensure accurate and efficient task planning.
- **Invoke external tools:** The framework employs various tools adeptly, such as the Python interpreter for mathematical computations or search engines for information retrieval, demonstrating the adaptability and practicality of the approach.
- **Validation:** Subsequent to the application of tools, the results undergo a thorough verification to ensure both the precision and effectiveness of the solutions. This procedure is imperative to preserve the quality of task planning. When an error is detected, the executor is informed of the error message and the subtask that needs to be re-executed.
- **Improvement:** The executor adds the error message to the prompt template and re-executes the tool call. To ensure that the program does not get stuck in an infinite loop, we set strict limits. Specifically, in phase validation and improvement, we limit the number of cycles to no more than five times and set the maximum cycle time for that phase to 5 minutes. In addition, we conduct repetitive checks on the content after each improvement to ensure that the model does not generate nonsense. If we encounter any of the following situations: The number of loops reaches the upper limit, the loop time exceeds 5 minutes, or the improved content has not changed from the previous phase, we terminate the current phase and proceed to the next phase. This is designed to ensure the stability and efficiency of the program.
- **Plan integration:** The verified solution is organized and presented as the final result of the task planning [27]. This output serves not only as a summary of the task-planning process but also as a response to the user's needs.

Through the implementation of this streamlined process, the LLM-Collab framework ensures that each phase of task planning is seamlessly integrated, culminating in an efficient and reliable method of task resolution.

3.2. Agent collaboration model

The two agents play the roles of analyst and executor. In our research, the task planning process is systematically divided into two levels: High-level planning and low-level planning. In high-level planning, we deploy the analyst role, which is responsible for in-depth analysis of the task

requirements described in natural language and breaking them down into several independent and executable subtasks that correspond to different phases of task planning. This process involves a macro-understanding of the task and strategic decision making. In low-level planning, the executor role assumes the responsibility for the specific implementation, operates and solves the independent subtasks of each phase, and provides the results to the analyst for further processing and integration. This hierarchical approach not only improves the efficiency and accuracy of task planning but also enhances the flexibility and extensibility of the whole system. The collaboration mechanism between them is central to complex task planning [28], as shown in Figure 4. The following is a detailed explanation of the collaboration mechanism between the two:

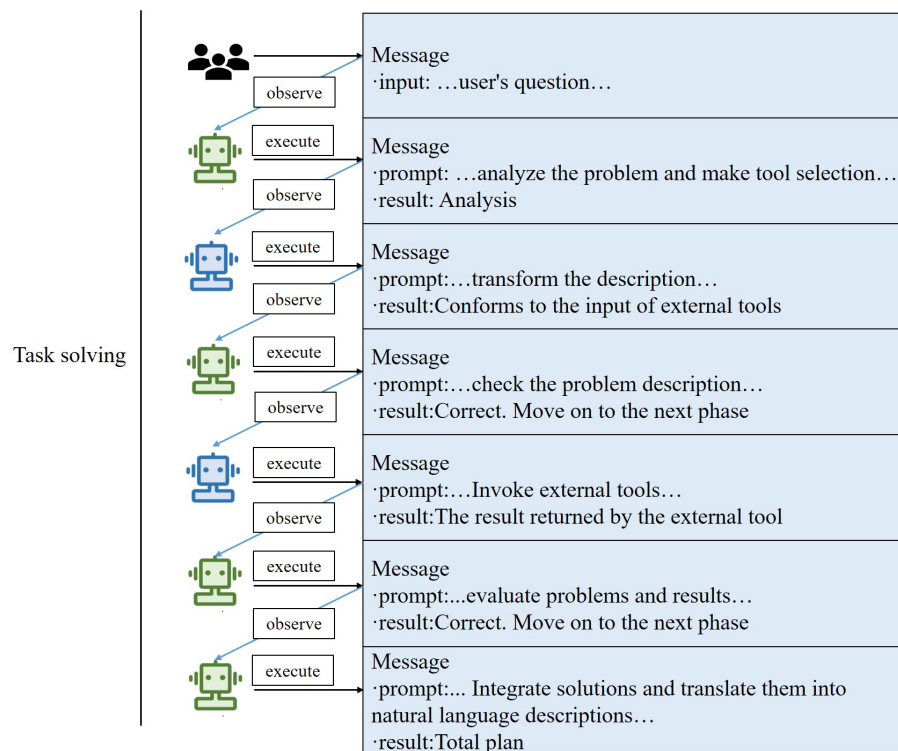


Figure 4. This illustration elucidates the procedure by which an agent reaches a decision, employing a series of interconnected cognitive chains that engage in sequential reasoning to ascertain the ultimate solution.

- **Analyst:** The analyst is responsible for managing the task planning process to ensure that the tasks are executed in accordance with the predetermined direction and steps. Using the chain-of-thought methodology, the analyst formulates intermediate reasoning phases to decompose complex problems into smaller, more manageable sections.
- **Executor:** To ensure efficient execution of specific subtasks, we follow these steps: First, we integrate the data provided by the analyst into a predefined prompt template. These templates are a framework for the structure of the text and specify the key information that the model needs to know, including the type of task, the input data, and the expected output format. We carefully structure the prompt template for each phase so that analysts and executors can fill in the relevant information accurately to generate specific and clear phase instructions. The executor will then

use the resources in the tool set to complete the task. This might involve utilizing a Python interpreter for complex mathematical calculations or employing a Baidu search engine to retrieve the necessary information, as well as incorporating any other available tools to accomplish the task effectively.

- **Prompt:** In the process of constructing the prompt, we employed a structure that integrates the a few-shot prompt with the chain of thought. This type of prompt is typically composed of three parts: instruction, rationale, and exemplars. Based on samples from the training set of the datasets used in the experiments, we meticulously designed the prompt to guide the model through intermediate reasoning steps to solve problems, thereby significantly enhancing the model's performance on complex reasoning tasks. In particular, the instruction describes the problem and indicates the expected format of the model's output. Rationale is the core of the chain of thought, encompassing the problem's solution, intermediate reasoning steps, and any relevant external knowledge. Exemplars exhibit to the model the fundamental format of input-output pairs. Each exemplar includes the problem, the reasoning process, and the answer.

Through multi-round communication and cooperation, the two agents gradually advance the solution of complex planning tasks [29]. The analyst's macro-control ensures the direction and consistency of task planning, while the executor's subtask resolution ensures the efficiency and accuracy of task execution. Their close cooperation automates and intelligently improves task planning [30].

3.3. External tool

In this study, we introduce a suite of external tools for LLMs, improving their capabilities and the efficiency and precision of task planning. These tools include:

- **Python interpreter:** The Python interpreter, which serves as an integral element of the toolkit, offers extensive programming functionalities. This capability enables models to perform intricate mathematical calculations, participate in logical reasoning processes, and handle data processing tasks with precision. The real-time execution of the Python code facilitates the verification of mathematical expressions, resolution of sophisticated mathematical challenges, and execution of both symbolic and numerical analyses, thus enhancing the analytical prowess of models in a diverse array of computational tasks. As shown in Figure 5.
- **Baidu search:** As an information retrieval tool in the toolkit, Baidu search empowers LLMs to access the vast information available on the Internet. Through search engines, models can quickly obtain up-to-date information and data in specific fields, crucial for understanding context, expanding knowledge boundaries, and answering queries.
- **PDDL planner:** The Planning Domain Definition Language planner is a structured task planning tool that plays a pivotal role in the toolkit for modeling and solving complex planning tasks. As a formal language, PDDL enables the logical and rigorous expression of task planning problems, which is crucial to automating planning and execution processes. PDDL is made of two major components: The domain definition and the problem instance. The domain definition outlines the predicates and actions that describe the rules of the planning environment, while the problem instance specifies the initial state, the goal, and the objects involved in a planning scenario. To address a variety of PDDL problems, a multitude of algorithms are employed, each bringing its

own unique approach to finding solutions. These algorithms may include classical planners that search for optimal sequences of actions to achieve a goal state from an initial state. They may also encompass temporal planners that account for time constraints within the planning process, as well as probabilistic planners that deal with uncertainty in planning outcomes. To ensure the accuracy and feasibility of the plans generated, we utilize the plan validation tool VAL, which collaborates with large language models by offering feedback on the outcomes of the plans crafted by the LLM, thereby enabling the model to fine-tune its strategies based on the validator's insights. Additionally, we employ the FAST-DOWNWARD planner for our planning tasks, a renowned classical planning system based on heuristic search that efficiently handles general deterministic planning problems encoded in the propositional fragment of PDDL. The versatility of PDDL supports these diverse planning paradigms, making it a robust language to express a wide range of planning problems and to leverage different algorithms to provide solutions. This comprehensive approach not only enables PDDL to effectively address complex tasks but also provides a versatile framework for artificial intelligence planning, facilitating the deployment of intelligent systems in practical devices and their interaction with the world [31].

An Example of Generate Executable Python Code Written by GPT-3.5

Prompt: Problem description + Problem analysis +
 Tool choice: python
 Current planning phase: Task Description Conversion.
 The problem description is converted into Python code to calculate the total running distance in a week. Return the python code for the plan.

GPT-3.5:

```
# Calculate James's total running distance per week
# Given parameters
distance_per_sprint = 60 # meters
sprints_per_week = 3
times_per_sprint = 3
# Calculate total distance
total_distance_per_week = distance_per_sprint * sprints_per_week
* times_per_sprint
# Print the result
print("Total meters James runs in a week:",
total_distance_per_week)
```

Figure 5. The generation of executable programs is achieved by inputting the specific problem and subsequently selecting the appropriate tool.

In our framework, the implementation of the tool call mechanism relies on a well-designed external tool list that not only contains the names of the available tools but also details the function and usage flow of each tool. The model can analyze the task requirements, automatically select the appropriate tool from this list, and generate the corresponding function calls. This process involves natural language parsing of the tool description and understanding the context of the task to ensure the correct selection and effective integration of the tool. In this way, LLM-Collab is able to dynamically expand its capabilities to accommodate various task requirements. Although LLMs effectively integrate

with widely used tools like the Python interpreter, they may lack the necessary sophistication when interfacing with professional tools such as traditional planners. We propose training specialized small models to handle tool invocation.

An innovative aspect of our approach is the provision of a highly extensible framework that can easily adapt to future technological developments by adding new tool names and descriptions to the list, thereby enhancing its planning capabilities and adaptability without fundamental revisions to the underlying methodology. This strategy not only streamlines the tool selection process but also ensures that the framework remains flexible and forward-compatible, enabling the seamless integration of emerging tools and technologies. By maintaining a dynamic and expandable tool list, we aim to equip the framework to evolve in tandem with the ever-changing landscape of AI and task planning.

4. Experiment

4.1. Benchmark problems

4.1.1. Mathematical reasoning

GSM8K [32], ASDiv [33], and SVAMP [34] are well-known datasets in mathematical reasoning. The GSM8K dataset focuses more on evaluating the model's multi-step reasoning ability; the ASDiv dataset emphasizes the diversity of problems and different difficulty levels; and the SVAMP dataset focuses more on the simple arithmetic problem solving ability. These three datasets provide rich test scenarios to evaluate and improve the mathematical capabilities of the models. The GSM8K dataset comprises 8,500 elementary mathematical problems involving fundamental arithmetic operations, which require 2 to 8 steps for resolution. It encompasses a training set of 7,500 problems and a test set of 1,000 problems. The answer to each GSM8K question includes a complete solution process to facilitate cot training. ASDiv datasets are diverse in terms of language patterns and problem types and are used to assess the model's ability to solve mathematical word problems. The SVAMP dataset comprises 1,000 mathematical word problems, each incorporating up to two mathematical expressions alongside a single unknown variable. In our experimental configuration, we selected test sets from three distinct datasets: the GSM8K dataset, which provided 1,319 problems; the ASDiv dataset, which provided 2,305 problems; and the SVAMP dataset, which provided 1,000 problems for evaluation. As shown in Table 1.

Table 1. Summary of math problem benchmarks with examples. N: number of evaluation examples.

Dataset	N	Example problem
GSM8K	1,319	Janet ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?
ASDiv	2,305	Seven red apples and two green apples are in the basket. How many apples are in the basket?
SVAMP	1,000	Each pack of dvds costs 76 dollars. If there is a discount of 25 dollars on each pack. How much do you have to pay to buy each pack?

4.1.2. Symbolic reasoning

We delve into the application of our method to symbolic reasoning tasks, which are critical for assessing an AI's ability to understand and reason with abstract concepts and relationships. Symbolic reasoning is exemplified through three diverse tasks from the BIG-Bench Hard dataset [35]: COLORED OBJECTS, PENGUINS, and DATE.

The COLORED OBJECTS task involves tracking and reasoning about the color and position of various objects, requiring the model to maintain a clear understanding of both relative and absolute positions. The PENGUINS task presents a more dynamic challenge, with a table of penguins and additional descriptive information, requiring the model to account for changes such as additions or removals of penguins and to answer attribute-related queries. The DATE task evaluates the model's ability to comprehend and perform operations on dates described in natural language, including computations involving time periods. In our experimental configuration, we extracted test subsets from three different datasets: The COLORED OBJECTS dataset, providing 250 problems; the PENGUINS dataset, providing 226 problems; and the DATE dataset, providing 319 problems for evaluation, as shown in Table 2.

Table 2. Summary of symbolic problem benchmarks with examples. N: number of evaluation examples.

Dataset	N	Example problem
COLORED OBJECTS	250	On the floor, there is one mauve cat toy, two purple cat toys, three grey cat toys, two mauve notebooks, three grey notebooks, three burgundy cat toys, and one purple notebook. If I remove all the notebooks from the floor, how many grey objects remain on it? Answer Choices: (a) one (b) two (c) three (d) four (e) five (f) six
PENGUINS	226	Here is a table where the first line is a header and each subsequent line is a penguin: name, age, height (cm), weight (kg) Louis, 7, 50, 11 Bernard, 5, 80, 13 Vincent, 9, 60, 11 Gwen, 8, 70, 15 For example: the age of Louis is 7, the weight of Gwen is 15 kg, and height of Bernard is 80 cm. What is the name of the last penguin sorted by alphabetic order? Answer Choices: (a) Louis (b) Bernard (c) Vincent (d) Gwen (e) James
DATE	319	Today is Christmas Eve of 1937. What is the date tomorrow in MM/DD/YYYY? Answer Choices: (a) 12/11/1937 (b) 12/25/1937 (c) 01/04/1938 (d) 12/04/1937 (e) 12/25/2006 (f) 07/25/1937

4.1.3. Robot planning tasks

Prior to engaging in robot planning tasks, it is essential to clarify the reasoning behind choosing tasks related to PDDL as the focal point of this study. Initially, PDDL-related tasks represent quintessential traditional planning tasks that are extensively utilized across academic and industrial domains. Additionally, as future agent development progresses toward tangibilization and deployment on diverse machines for real-world interaction, scrutinizing the efficacy of these agents in planning

tasks is of paramount importance. In this section, we evaluate our approach to addressing robot planning tasks utilizing a comprehensive set of benchmark problems.

This paper uses a suite of benchmark problems to assess our method for addressing robot planning tasks. Developed from past International Planning Competitions, these benchmarks include seven distinct robot planning domains, each featuring 20 automatically generated tasks with natural language descriptions and the corresponding PDDL files. The domains include BLOCKSWORLD, BARMAN, FLOORTILE, GRIPPERS, STORAGE, TERMES, and TYREWORLD. Each domain is equipped with an example problem description, a PDDL file, and a plan description that serve as context for various approaches. It is assumed that each problem domain is furnished with a domain PDDL file provided by the user or a domain expert.

4.2. Experiment setup

We conducted experiments on three categories of tasks to evaluate the performance of our task planning method. The first category involved mathematical problems; we utilized the GSM8K, SVAMP, and ASDiv datasets to assess the model's ability to solve a diverse set of mathematical challenges. The second category was symbolic reasoning; we selected challenging problems from the BIG-Bench Hard dataset to evaluate the model's capacity for symbolic reasoning. The third category was robot planning tasks. We tested robot planning tasks in seven fields using the FAST-DOWNWARD planner [36]. In the third experiment, we used the SEQ-OPT-FDSS-1 algorithm, an optimized instance of the FAST-DOWNWARD Stone Soup framework, to ensure optimal solutions. This algorithm is distinguished by its ability to generate sequence-optimal plans and its support for STRIPS expressivity, a foundational planning language in artificial intelligence. STRIPS is recognized for its capacity to define problems using atomic propositions, serving as a key component in automated planning and paving the way for more advanced planning languages like PDDL. Using this algorithm, we were able to identify the lowest-cost solutions within specified constraints. With a maximum search time of 200 s. We measured the optimal planning success rate for these tasks.

We explored LLM-Collab on multiple benchmarks. Next, we explain the symbols utilized in the control experiments to provide clarity on the setup and interpretation of the comparative trials. As shown in Table 3.

- **Direct:** We consider standard few-shot prompting, popularized by Brown et al. [37], in which a language model is given in-context exemplars of input-output pairs before outputting a prediction for a test-time example. Exemplars are formatted as questions and answers. The model gives the answer directly, as shown in Figure 1 (left).
- **CoT:** The cot approach is to augment each exemplar in few-shot prompting with a chain of thought for an associated answer, as shown in Figure 1 (right).
- **LLM-Collab:** Throughout this process, the content within the prompt templates is carefully customized to meet the specific demands of each stage, ensuring a seamless integration of inputs for the establishment of cognitive chain templates. In each round of dialogue, the analyst first conducts an assessment of the current task status, which involves reviewing the task's progress and identifying which components have been completed and which require additional attention. Following this assessment, the analyst offers feedback on the subtasks executed by the executor, including validating the results, pinpointing any errors or omissions, and suggesting areas for improvement. Using these evaluations and feedback, the analyst and executor collaboratively

make program improvements, which may include adjusting strategies, selecting new tools or methods, or refining the plan.

- **LLM-:** The LLM- approach submits test problems described in natural language directly to large language models without any contextual information, requiring the model to output a planning solution described in natural language.
- **LLM:** The LLM method feeds test problems described in natural language into LLM, accompanied by contextual information, and also demands that the model produce a planning solution articulated in natural language.
- **LLM+P:** The LLM+P method is a framework that integrates the strengths of classical planners into LLMs. It converts planning problems described in natural language into Planning Domain Definition Language files, leverages classical planners to quickly find solutions, and then translates the solutions back into natural language [8]. This approach enables LLMs to solve complex planning problems without altering the models themselves.

Table 3. Prompt templates for solving problems.

Experiment	Prompt Template
Direct	{examples} {question}
CoT	{cot examples} {question}
LLM-Collab	You are a {persona}. Your task is {planning task}, and you can { instruction}. {cot examples} {question} {instruction}
LLM-	{question}
LLM	You are a {persona}. You have {tools}, and you can do:{actions} You have the following restrictions on your actions:{restrictions} Now consider a planning problem. The problem description is:{description} {question} Can you provide an optimal plan, in the way of a sequence of behaviors, to solve the problem?
LLM+P	You are a {persona}. You have {tools}, and you can do:{actions} You have the following restrictions on your actions:{restrictions} Now consider a planning problem. The problem description is:{description} {question} Provide me with the problem PDDL file that describes the planning problem directly without further explanations.

In each experimental group, we integrated the problems with the reasoning chain templates introduced in Section 3, feeding them into the model for testing without any modifications to the problems. The experiment is designed to compare the performance of our proposed method with the baseline method. The experimental procedures were executed using the capabilities of the GPT-3.5 turbo model. The evaluation measure was the correctness of the solutions provided by the model. Mathematical and logic questions were compared with standard answers to determine the correct rate. The validity of the robot planning task was judged by the VAL validator. To ensure that the program did not get stuck in an infinite loop, we set strict limits. Specifically, in phase validation and improvement, we limited the number of cycles to no more than five times and set the maximum cycle time for that

phase to 5 minutes. In addition, we conducted repetitive checks on the content after each improvement to ensure that the model does not generate nonsense. If we encountered any of the following situations: The number of loops reached the upper limit, the loop time exceeded 5 minutes, or the improved content did not change from the previous phase, we terminated the current phase and proceeded to the next phase. This is designed to ensure the stability and efficiency of the program.

4.3. Results and analysis

4.3.1. Mathematical reasoning results

Figure 6 presents the results of our experiments, which were divided into three groups. The first group involved the model directly outputting the results, the second group utilized the CoT strategy, and the third group employed the LLM-Collab strategy. The experimental results demonstrate that the CoT strategy significantly enhances the model's accuracy in solving mathematical reasoning tasks. However, the inherent computational limitations of LLMs can lead to incorrect results when addressing multi-step mathematical problems. The LLM-Collab strategy, which incorporates an external Python interpreter, equips LLMs with enhanced computational capabilities. Furthermore, collaboration between two agents significantly improves the computational accuracy.

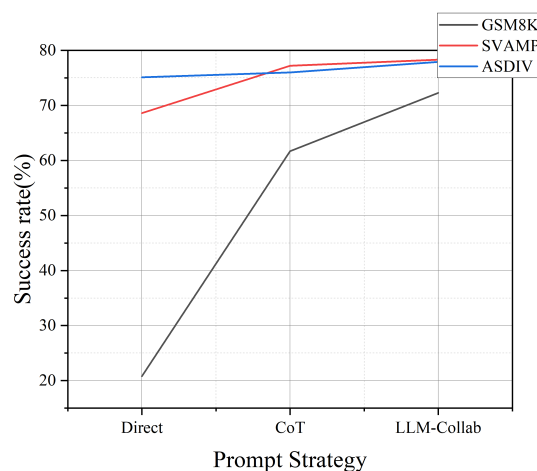


Figure 6. In mathematical reasoning, three different inputs are used to test the accuracy.

4.3.2. Symbolic reasoning results

Figure 7 presents the success rates of symbolic reasoning tasks. We evaluated the performance of the model across three experimental setups. The results underscore the effectiveness of the CoT strategy in increasing the model success rate across the three datasets. In particular, the LLM-Collab strategy achieves the highest success rates, demonstrating a significant improvement over the other methods. This outcome highlights the potential of collaborative approaches to overcome the computational limitations inherent in LLMs when tackling complex reasoning tasks.

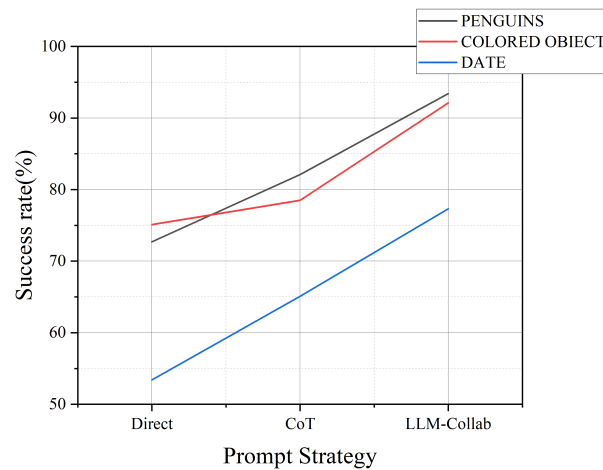


Figure 7. In symbolic reasoning, three different inputs are used to test the accuracy.

4.3.3. Robot planning tasks results

Figure 8 shows the following results: In this research, we conducted an in-depth analysis of the LLM+ model's ability to formulate plans for problems presented in a natural language environment. Despite the effort of the model to provide solutions, most of the generated plans were not viable in practical applications. The root cause lies in the obvious shortcomings of the LLM+ model in logical reasoning, particularly in analyzing the prerequisite conditions of the problem. We observed consistent failure patterns in the model when dealing with problems, regardless of whether sample plans were provided as references. To address this issue, we replicated the LLM+P method and found that it could provide optimal solutions for most problems. However, if key initial conditions were incorrectly specified in the problem file, such as omitting the condition for tile breakage, the planning problem could not be effectively resolved. Furthermore, we found that without context information, i.e., the absence of an example problem and its corresponding problem PDDL file, the LLM model was unable to correctly generate a problem PDDL file. This indicates that context information is crucial for the efficient operation of the LLM+P method. Through comparative analysis, we found that the LLM-Collab model, through agent collaboration, could correct some complex logical relationships, but it has limitations in understanding and processing complex spatial relationships, often leading to task failure.

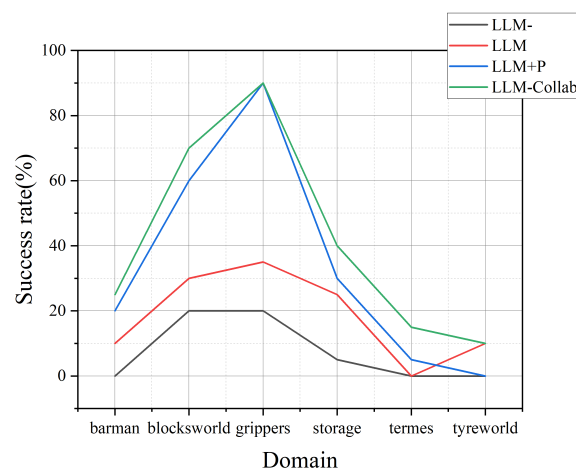


Figure 8. In robot planning tasks, four different inputs are used to test the accuracy.

4.3.4. Context samples and model performance

To further investigate the efficacy of the proposed method, the number of context samples and test questions was systematically altered, and their impacts were analyzed. Figure 9 illustrates the dynamic trend in the precision of the LLM-Collab model with a progressive increase in the number of test questions. It was observed that increasing the number of context samples substantially improves model performance. However, when the number of context samples reaches a certain level, the incremental performance gains from additional samples decrease. Furthermore, an increase in sample size may also incur greater costs, necessitating careful consideration of the cost-benefit ratio. Concurrently, it is recognized that the quality of context samples is more critical to model performance than their mere quantity. By meticulously designing each context sample to be rich in valid information, we can significantly enhance the model's ability to generalize and its accuracy. Therefore, we advocate for a focus on improving the quality of the samples instead of simply increasing their quantity during model training. Particularly when tasks require multi-step reasoning or are not well-represented in the LLMs' training data, our framework may struggle with extremely complex tasks that demand sophisticated reasoning or domain-specific knowledge beyond the current LLMs' capabilities, which could result in less than ideal performance. Moreover, our framework's performance heavily relies on the quality and quantity of data used for context-prompting LLMs, and in cases where the dataset is small or biased, the framework may not generalize well to new, unseen problems.

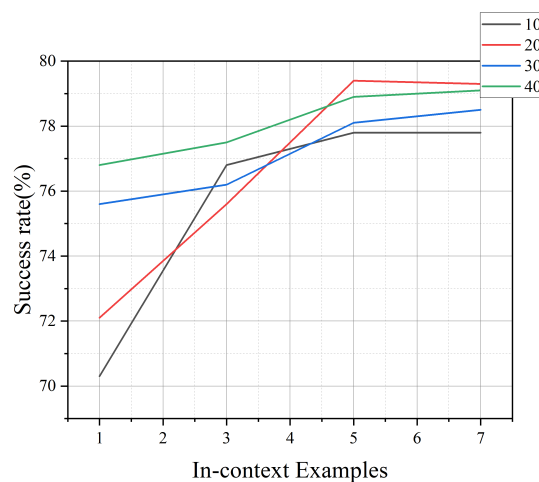


Figure 9. Accuracy for varying numbers of in-context examples and training examples.

Our experiments substantiated the efficacy of our approach in complex task planning scenarios. In mathematical problem solving, our method achieved a high success rate using datasets adeptly to tackle diverse mathematical challenges. Notably, our structured methodology significantly enhanced the model's reasoning capabilities in the BIG-Bench Hard symbolic reasoning tasks. Particularly impressive was the method's consistent identification of optimal solutions within time constraints in robot planning tasks, surpassing the baseline manual approach.

5. Conclusions

Through the integration of cot reasoning with LLM, we formulated a method that enhances the efficiency, adaptability, and precision of AI in task planning in diverse environments. Our comprehensive approach, which extends from theoretical foundations to empirical validation, has resulted in a robust framework that effectively addresses the complexities inherent in real-world planning challenges.

The empirical findings highlight the efficacy of our approach. Elevated success rates, optimization of resource usage, and scalability substantiate the advantages of our method compared to conventional planning techniques. Moreover, the comprehensive exposition of the task planning process through the thought chain enhances the system's explicability.

Our work offers several key contributions: First, we introduce an innovative integration of cot reasoning with LLMs, resulting in a novel approach to task planning; second, we present a comprehensive framework that facilitates the analysis and modeling of diverse task environments; and third, we substantiate the efficacy of our methodology through empirical validation against recognized benchmarks.

Although our method has shown strong performance, it requires refinement in the integration of cot with LLM, generalization of the findings in more scenarios, and improvement of user interaction and personalization. High-quality and highly relevant prompt samples are instrumental in enhancing the performance of the model. Researchers should focus on improving CoT-LLM integration for efficiency, broadening the scope of tasks and environments for generalizability, improving user interaction for personalization, and evaluating real-world deployment for practical utility. The goal is to develop a smarter, more flexible task planning system, with ongoing efforts to combine advanced reasoning techniques with AI models for enhanced capabilities.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools to generate the content of this article, but some AI based tools were used for grammar checking and sentence optimization.

Conflict of interest

Yanlong Zhai and Jun Shen are the editorial board members for Applied Computing and Intelligence and were not involved in the editorial review and the decision to publish this article.

References

1. J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, M. S. Bernstein, Generative agents: interactive simulacra of human behavior, *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023, 1–22. <https://doi.org/10.1145/3586183.3606763>
2. H. Yang, S. Yue, Y. He, Auto-gpt for online decision making: benchmarks and additional opinions, arXiv: 2306.02224. <https://doi.org/10.48550/arXiv.2306.02224>

3. Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, et al., A survey on in-context learning, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, 1107–1128. <https://doi.org/10.18653/v1/2024.emnlp-main.64>
4. Z. Wang, Z. Cheng, H. Zhu, D. Fried, G. Neubig, What are tools anyway? a survey from the language model perspective, arXiv: 2403.15452. <https://doi.org/10.48550/arXiv.2403.15452>
5. C. Aeronautiques, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, et al., Pddl— the planning domain definition language, Tech. Report CVC TR-98-003/DCS TR-1165, 1998.
6. J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, et al., Deep reinforcement learning with a natural language action space, arXiv: 1511.04636. <https://doi.org/10.48550/arXiv.1511.04636>
7. Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, et al., The rise and potential of large language model based agents: a survey, *Sci. China Inform. Sci.*, in press. <https://doi.org/10.1007/s11432-024-4222-0>
8. B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, et al., Llm+p: empowering large language models with optimal planning proficiency, arXiv: 2304.11477. <https://doi.org/10.48550/arXiv.2304.11477>
9. L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, et al., Pal: program-aided language models, *Proceedings of the 40th International Conference on Machine Learning*, 2023, 10764–10799.
10. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, et al., React: synergizing reasoning and acting in language models, *Proceedings of 11th International Conference on Learning Representations, ICLR*, 2023, 1–33.
11. J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, et al., Chain-of-thought prompting elicits reasoning in large language models, *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2024, 24824–24837.
12. R. Prabhakar, R. Sivaramakrishnan, D. Gandhi, Y. Du, M. Wang, X. Song, et al., Sambanova sn40l: scaling the ai memory wall with dataflow and composition of experts, *Proceedings of 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024, 1353–1366. <https://doi.org/10.1109/MICRO61859.2024.00100>
13. A. Talmor, J. Herzig, N. Lourie, J. Berant, CommonsenseQA: a question answering challenge targeting commonsense knowledge, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, 4149–4158. <https://doi.org/10.18653/v1/N19-1421>
14. M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, et al., Challenging BIG-bench tasks and whether chain-of-thought can solve them, *Proceedings of Findings of the Association for Computational Linguistics: ACL 2023*, 2023, 13003–13051. <https://doi.org/10.18653/v1/2023.findings-acl.824>
15. N. Crispino, K. Montgomery, F. Zeng, D. Song, C. Wang, Agent instructs large language models to be general zero-shot reasoners, *Proceedings of the 41st International Conference on Machine Learning*, 2024, 9458–9549.

16. X. Huang, W. Liu, X. Chen, X. Wang, H. Wang, D. Lian, et al., Understanding the planning of LLM agents: a survey, arXiv: 2402.02716. <https://doi.org/10.48550/arXiv.2402.02716>
17. F. F. Xu, Y. Song, B. Li, Y. Tang, K. Jain, M. Bao, et al., Theagentcompany: benchmarking llm agents on consequential real world tasks, arXiv: 2412.14161. <https://doi.org/10.48550/arXiv.2412.14161>
18. Y. Chen, W. Wang, S. Lobry, C. Kurtz, An llm agent for automatic geospatial data analysis, arXiv: 2410.18792. <https://doi.org/10.48550/arXiv.2410.18792>
19. C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, et al., Chatdev: communicative agents for software development, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024, 15174–15186. <https://doi.org/10.18653/v1/2024.acl-long.810>
20. J. Zhao, C. Zu, H. Xu, Y. Lu, W. He, Y. Ding, et al., Longagent: scaling language models to 128k context through multi-agent collaboration, arXiv: 2402.11550. <https://doi.org/10.48550/arXiv.2402.11550>
21. P. Gong, J. Li, J. Mao, Cosearchagent: a lightweight collaborative search agent with large language models, *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, 2729–2733. <https://doi.org/10.1145/3626772.3657672>
22. X. Feng, Z. Y. Chen, Y. Qin, Y. Lin, X. Chen, Z. Liu, et al., Large language model-based human-agent collaboration for complex task solving, arXiv: 2402.12914. <https://doi.org/10.48550/arXiv.2402.12914>
23. P. S. Dhillon, S. Molaei, J. Li, M. Golub, S. Zheng, L. P. Robert, Shaping human-ai collaboration: varied scaffolding levels in co-writing with language models, *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, 1–18. <https://doi.org/10.1145/3613904.3642134>
24. J. Oswald, K. Srinivas, H. Kokel, J. Lee, M. Katz, S. Sohrabi, Large language models as planning domain generators, in *Proceedings of the International Conference on Automated Planning and Scheduling*, **34** (2024), 423–431. <https://doi.org/10.1609/icaps.v34i1.31502>
25. C. H. Song, J. Wu, C. Washington, B. M. Sadler, W. L. Chao, Y. Su, Llm-planner: few-shot grounded planning for embodied agents with large language models, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, 2998–3009.
26. Z. Jiao, Y. Niu, Z. Zhang, S. C. Zhu, Y. Zhu, H. Liu, Sequential manipulation planning on scene graph, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, 8203–8210. <https://doi.org/10.1109/IROS47612.2022.9981735>
27. Y. Jiang, S. Zhang, P. Khandelwal, P. Stone, Task planning in robotics: an empirical comparison of pddl-and asp-based systems, *Frontiers Inf. Technol. Electronic Eng.*, **20** (2019), 363–373. <https://doi.org/10.1631/FITEE.1800514>
28. B. Y. Lin, Y. Fu, K. Yang, F. Brahman, S. Huang, C. Bhagavatula, et al., Swiftsage: a generative agent with fast and slow thinking for complex interactive tasks, *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2024, 23813–23825.

29. J. Xu, H. Wang, Z. Y. Niu, H. Wu, W. Che, Knowledge graph grounded goal planning for open-domain conversation generation, *Proceedings of the AAAI Conference on Artificial Intelligence*, **34** (2020), 9338–9345. <https://doi.org/10.1609/aaai.v34i05.6474>
30. S. Agashe, Y. Fan, A. Reyna, X. E. Wang, Llm-coordination: evaluating and analyzing multi-agent coordination abilities in large language models, 2024, arXiv: 2310.03903. <https://doi.org/10.48550/arXiv.2310.03903>
31. P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, *An introduction to the planning domain definition language*, Cham: Springer, 2019. <https://doi.org/10.1007/978-3-031-01584-7>
32. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, et al., Training verifiers to solve math word problems, arXiv: 2110.14168. <https://doi.org/10.48550/arXiv.2110.14168>
33. S. Y. Miao, C. C. Liang, K. Y. Su, A diverse corpus for evaluating and developing english math word problem solvers, arXiv: 2106.15772. <https://doi.org/10.48550/arXiv.2106.15772>
34. A. Patel, S. Bhattamishra, N. Goyal, Are NLP models really able to solve simple math word problems? *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, 2080–2094. <https://doi.org/10.18653/v1/2021.naacl-main.168>
35. A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoen, A. Abid, A. Fisch, et al., Beyond the imitation game: quantifying and extrapolating the capabilities of language models, arXiv: 2206.04615. <https://doi.org/10.48550/arXiv.2206.04615>
36. M. Helmert, The fast downward planning system, *J. Artif. Intell. Res.*, **26** (2006), 191–246. <https://doi.org/10.1613/jair.1705>
37. T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, et al., Language models are few-shot learners, *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, 1877–1901.



AIMS Press

©2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0>)