ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 3 - 2026

A Graph-Based and Large Language Model-Driven Task Planning and Scheduling Method for Intelligent Agents

Huanyu Cheng[1*], *Yingcheng Gu*[1], *Mengting Xi*[1], *Qiuyuan Zhong*[1] *and Wei Liu*[1]

[1] State Grid Jiangsu Electric Power Co., Ltd. Information and Communication Branch, Nanjing China

SUMMARY: *Task planning is important in intelligent proxy systems to help agents turn advanced goals into effective operations. But now methods often face problems with problem expansion, modeling dependency, and open environment stability. To address these problems, this article offers a new planning and planning structure that combines large language models (LM) with graphical reasoning. After providing a description of tasks in natural language, LLM will disassemble it into simple subtains, as well as determine local priority relationships, which will then become global dependencies. The results of the subtaspecty were compared, causing targeted self-correcting. Multi-round verifiability tests ensured the reliability of the problem of the sub-tasting task. Experiments on a multi-examination dataset set of 100 samples showed that this method worked much better than the baseline, and reflection-based approach, indicating that it has the potential to create stable and interpreted intelligent agents.*

KEYWORDS: *Task planning, Big language models, Intellectual agents, Graphic modeling, Reflection, Planning*

1 Introduction

In recent years, large language models (LLM), such as ChatGPT, Claude, Gemini, LLama, Qwen and DeepSeek, have significantly succeeded in working with natural language, reasoning, content generation and multimodal understanding[1–6]. Their Semantic knowledge and reasoning capabilities, trained through a large body, were used in many places, such as answers to questions, code generation, mathematical problems[7–9], customer service, etc. D. In the main AI system, task planning is important in the “perception-cognitive-action” cycle[10, 11]. It can determine whether an agent can turn an external goal into an internal set of actions[12]. Mission planning is crucial for the autonomy of the agent, reactivity and purposeful behavior – it generates a series of steps that can be made so that the agent can achieve the desired goal in a dynamic environment[13]. LLM has several natural advantages in task decomposition, planning, strategy evaluation, and error correction[14]. Now the study has five main directions: the decomposition of the task, the first is the decomposition of the task, which is to break complex tasks into smaller, well-treated subtasks[15–18]. The second is the multiplanning of generation and selection, which is to create several candidate programs by searching or evaluating, and then select the best[7,19]. Thought wood toT will create a “tree of thought” to study various plans; the self-inconsistency method will be the answer of the majority of votes; LLM-MCTS will combine LLM and the search for Monte Carlo trees to guide decision-making

*chenghuanyu2000@163.com
https://doi.org/10.65102/is20261229
Published: 10 June, 2026

[20-29]. The third is to use the external planner to generate formal representations (such as PDDL) "Reflection" and Self-Refine use feedback loops to improve the plan; CRITIC uses an external knowledge base to improve the ability to reflect. The fifth is improved memory planning, which is the knowledge and history of memory module tasks before storing memory[30-38]. This limits their use in high-risk areas such as energy, finance and healthcare.

To solve these problems, this article proposes an approach to problem planning for intelligent agents, which combines a graphical structure with large language models (LLM). Starting with a description of tasks in natural language, the method uses LLM to break the value of the task, as well as to extract the relationship between tasks. These subtacis and priorities are then presented by graphic structures so that task charts can be made, and the extended premine (EPM) can be used to guide the agent to effectively perform tasks – sometimes sequentially, sometimes in parallel. We also add a double-pout mechanism to compare the direct and indirect results of the direct and indirect sub-taste of LLM by extending the front[21, 22]. This contrast reflection of a non-core evaluator allows LLM to change the output when the results are inconsistent and significantly improve the stability of the plan. There are also methods that are reflected or self-corrected by external appraisers, or LLM's own offerings. Our method is more focused and better understood by reflection, transparency of tasks and agency, such as multi-wheeled, and the algorithm is well designed, combining the possibilities of generalizing LLM and rigor graphic modeling. Unlike traditional approaches, this work represents the first attempt to integrate LLM reasoning with graph-based dependency computation, offering a novel and interpretable framework for building highly reliable intelligent agents.

The remainder of this paper is organized as follows: Section 2 introduces the proposed task planning method based on graph structures and large language models; Section 3 presents the experimental setup and results; and Section 4 concludes the paper and outlines directions for future research.

2 Task Planning Method for Intelligent Agents Based on Graph Structures and Large Language Models

The overall framework of the proposed task planning method, which integrates graph structures and large language models (LLMs), is illustrated in Figure.1 as below.

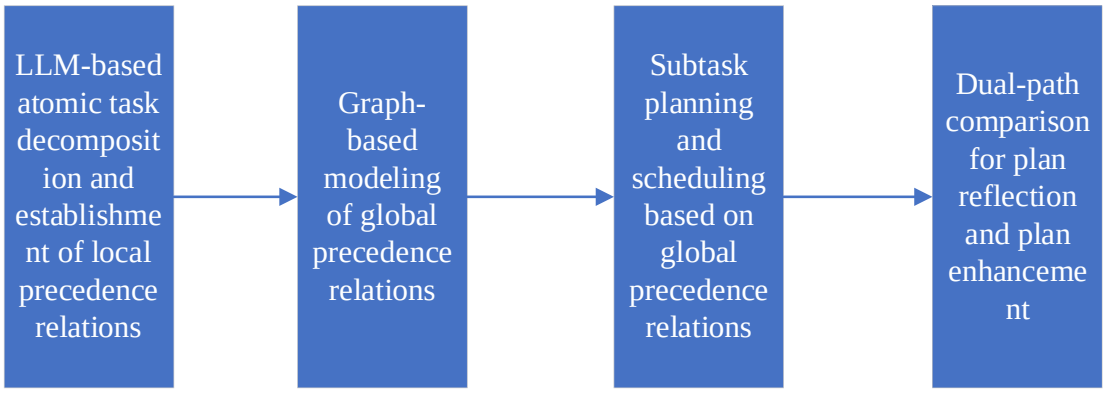

Figure 1: The framework of our method

The framework can be broadly divided into four stages: 1). Atomic task decomposition and local precedence relation extraction using LLMs; 2). Global precedence modeling based on graph structures; 3). Subtask planning and scheduling based on the global precedence matrix;

4). Dual-path reflection and optimization through comparison between direct and structured planning outputs.

In the first step, the LLM is used to decompose the user-submitted task into atomic subtasks and identify their immediate precedence relationships. The second step uses these local dependencies to build a graph-based representation that models global precedence relations among the subtasks. In the third step, task scheduling is performed based on this global precedence structure. The fourth step compares the final results set directly with the large language model with the results of step-in sub-site tasks. If they are very different, review, change and improve the previous plan.

2.1 Atomic Task Decomposition and Local Precedence Relation Extraction Using LLMs

This step, like many existing task-plant methods based on a large language model (LLM), builds on language understanding and the possibilities of reasoning LLM itself to solve complex problems, and can also find a connection between these small tasks. We make a hint B, which is: “Pleas, break the following problems into several subtakas that cannot be eliminated, and tell me what problem you need to solve first for each subtay, and then add A. The prompt B is then passed to the LLM embedded in the agent, which returns a structured task decomposition.

For example, consider the user-submitted question:“Wade Watts is the uncle of a politician. For which university did this politician play football?” [39].The constructed prompt would be: “Please break down the following question into several indivisible sub-questions, number them, and specify the prerequisite question(s) for each one: ‘Wade Watts is the uncle of a politician. For which university did this politician play football?’”Using the Qwen3 model, the LLM returns the following:

Sub-question 1: Who is Wade Watts?

Prerequisite: None (this is a basic information retrieval question)

Sub-question 2: Does Wade Watts have a nephew or niece who is a politician?

Prerequisite: Sub-question 1 (requires knowledge of Wade Watts' identity to explore his family relationships)

Sub-question 3: What is the profession of this politician?

Prerequisite: Sub-question 2 (requires confirmation of the familial relationship before exploring the politician’s profession)

Sub-question 4: Has this politician ever played football?

Prerequisite: Sub-question 3 (once the profession is known, the next step is to determine any athletic background, particularly football)

Sub-question 5: If the politician indeed played football, for which university did they play?

Prerequisite: Sub-question 4 (after confirming football involvement, the final step is to identify the specific university)

2.2 Global Precedence Modeling Based on Graph Structures

Let the set of decomposed atomic tasks be denoted as $\{A_1,A_2,\ldots A_n\}$. A local precedence matrix $\boldsymbol{PM}$ is then constructed, where each element is defined as:

$$\boldsymbol{PM}_{i,j}=\begin{cases}1, & i\neq j\wedge i\in \boldsymbol{Predecessors}(j)\\ 0, & i\neq j\wedge i\notin \boldsymbol{Predecessors}(j)\\ 0, & i=j\end{cases}\qquad 1\leq i,j\leq n \tag{1}$$

where ***Predecessors(j)*** denotes the set composed of all the predecessor subtasks of subtask A_j.

The method for calculating the global predecessor matrix ***EPM*** based on the predecessor matrix ***PM*** is as follows:

$$\boldsymbol{EPM}=\mathbf{sign}\left(\sum_{i=1}^{n}\boldsymbol{PM}^{i}\right) \tag{2}$$

where $\boldsymbol{PM}^i$ denotes the i-th power of matrix ***PM***. During computation, when $\boldsymbol{PM}^i=\mathbf{0}$, all subsequent$\boldsymbol{PM}^{i+1},\ldots\boldsymbol{PM}^n$ will be equal to the n-order zero matrix and need not be calculated. Then, set all elements greater than 0 in the matrix sum to 1. As in the example from the previous section, $\boldsymbol{PM}=\begin{pmatrix}0&0&0&0&0\\1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\end{pmatrix}$ After computation,$\boldsymbol{EPM}=\begin{pmatrix}0&0&0&0&0\\1&0&0&0&0\\1&1&0&0&0\\1&1&1&0&0\\1&1&1&1&0\end{pmatrix}$

It can be observed that the local precedence matrix (PM) only captures immediate relationships: task A_1 is a prerequisite for A_2, A_2 for A_3, A_3 for A_4, and A_4 for A_5. However, the extended precedence matrix (EPM) captures transitive dependencies: A_1 is a prerequisite for A_2; A_1 and A_2 for A_3; A_1, A_2, A_3 for A_4; and A_1, A_2, A_3, A_4 for A_5. In other methods that rely solely on LLM-based reasoning, such transitive dependencies (e.g., between A_1 and A_3) may not be identified. As a result, subtasks like A_1 and A_3 may be executed in parallel erroneously, leading to incorrect outcomes. The study uses A_1 modeling as a prerequisite for A_3 to determine its priority after A_1.

2.3 Subtask Planning and Scheduling Based on Global Precedence Relations

The proposed subtask planning and scheduling algorithm is detailed below:

Algorithm 1: Subtask Planning and Scheduling Based on the Global Precedence Matrix

Input: Subtask sequence $A_1,A_2,\ldots A_n$, extended precedence matrix EPM

Output: task result, executed subtask list *LASK*, and their respective outputs

Step 1: Initialize the result set and task execution list as $S_1=\varnothing,S_2=\{A_1,A_2,\ldots A_n\}$, $LASK=[],LANS=[]$.

Step 2: If $S_1\neq S_2$, then $SEPM_i=\sum_j EPM_{i,j}$, $1\leq i,j\leq n$. When $S_1=S_2$, it means that all subtasks have been completed and then this procedure has ended.

Step 3: For each unexecuted subtask A_i, check whether all its prerequisite tasks have been completed by evaluating the corresponding column in the EPM. If so, submit A_i to the LLM agent for execution in parallel. Record the result as R_i, append A_i to LASK list and set the entire i-th column of the EPM to zero. The former operations can be represented as $S_1=S_1\cup\{A_i\}$,$LASK=LASK.Append(A_i)$, $LANS=LANS.Append(R_i)$. Then return to Step 2.

Step 4: If current subtask is not executable which can be represented as $EPM_i\neq 0$, then increment the index i as $i=i+1$. If $i<=n$, then continue to Step 3. Otherwise, wait until any in-progress subtask completes, then return to Step 2.

This algorithm ensures that a subtask can only be executed when all of its prerequisite tasks have been completed, thus guaranteeing the correctness of execution. The research will prioritize the execution of subtasks that have no dependencies or have completed prerequisites, in order to achieve the highest level of parallelism.

2.4 Dual-Path Reflection and Planning Optimization

After all the sub-there are all the tasks, we are still not sure that the result is correct. It is unrealistic and expensive to check that each result is unrealistic, and some practical scenarios can't do it at all. To solve this problem, we use correlation feedback and pseudo-relevant feedback. In particular, we will compare the final results obtained with the help of the sub-trick, step by step, and directly allow the model of the large language (LLM) to solve the whole problem.

The dual-path reflection and optimization algorithm is formalized as follows:

Algorithm 2: Task Planning Reflection and Optimization

Input: user-submitted task *A*, Subtask sequence $A_1, A_2, \ldots A_n$

Output: final task result

Step 1: Submit the overall task *A* directly to the LLM-based agent to obtain the answer *Answer2*, and map it to an embedding vector $embedding_{A2}$ using an embedding model.

Step 2: Let *Answer1* be the result obtained by executing all subtasks. Map it to a vector $embedding_{A1}$, and calculate the similarity between $embedding_{A1}$ and $embedding_{A2}$ as $simi_{12}=\frac{embedding_{A1}\cdot embedding_{A2}}{|embedding_{A1}|\cdot|embedding_{A2}|}$.

Step 3: If $simi_{12}\geq T$, return *Answer1* to the user as the final result.

Step 4: If $simi_{12}<T$, then construct the following prompt: "For the question (" + *A* + "), when decomposed into a sequence of subtasks executed in order, the answer is (" + *Answer1* + "); whereas answering directly yields (" + *Answer2* + "). Why do the two answers differ? Please reflect step by step. Note: You may use the following information: the execution sequence of subtasks is " + [subtask order] + ", and the results of the subtasks are: " + [subtask outputs] + ".", Submit this prompt to the LLM.

Step 5: Then construct a new prompt: "Based on the above reflection, please re-answer the following question: " + *A*,Submit the prompt to the LLM and return the result to the user.

In the above algorithm, we set the value of *T* as 0.8. This algorithm leverages multi-path validation and LLM reflection to improve result accuracy. Unlike simple reflection mechanisms, our approach is clear-tell a large language model, exit, and input part of each subtaspecial puzzle to a large language model so that it can add context-hazardous reflections to provide enough information to reflect. If the results of the two paths are similar, the model is very confident in the results, and you do not need to think about it. Because our system combines large language models, external tools and memory modules and is reflected in the most important interactive records, so that the answers are more appropriate.

The following algorithms are proposed to reduce risks, and the validation of subtask outputs is as follows:

Algorithm 3: Subtask Output Verification Algorithm

Input: Subtask A_i

Output: Verified answer $Answer_i$ of Subtask A_i

Step 1: Construct the prompt: "Please answer the following question in a complete sentence: " + A_i, Submit it to the LLM to obtain $Answer_i$.

Step 2: Construct a rephrasing prompt:"Please paraphrase the following sentence without changing its meaning. Return only one alternative, no extra text: " +$Answer_i$, Submit to the LLM to obtain $Answer_i'$

Step 3: Formulate a verification prompt as *A'*: "Please answer the following question with only 'Yes', 'No', or 'Uncertain', and nothing else: " + $Ques_i'$. To generate $Ques_i'$, replace the ending punctuation of $Answer_i'$ with a question mark, or append a question mark if none is present.

Step 4: Submit the verification prompt *A'* to the LLM to obtain its answer denoted as

T_Ans.

Step 5: If *T_Ans* is 'Yes', then accept $Answer_i$ as the final answer and return it. Otherwise, construct the following reflective prompt as *T_Prompt*: "The answer to the question [" + A_i + "] is given as [" + $Answer_i$ + "], but when asked 'Please answer with Yes, No, or Uncertain: " + $Ques_i$' + "', the response was [" + *T_Ans* + "]. Please reflect on the inconsistency and improve."

Step 6: Submit the reflective prompt to the LLM and return to Step 1.

This verification cycle allows a large language model (LLM) to pass another term, check it yourself, and then confirm that the answer is inoperable. If the answer is found controversial, remind the models to think about it and change the answer. In order not to keep the loop from the fact that the cycle did not go, we let the maximum number of re-examinations-to this number the loop stops.

3 Experiments

3.1 Dataset

We have made a dataset of 100 multi-op-inferion tasks that are generated by LLM and are used for evaluation. Since our methods don't require training, we only need to check the data. These issues cover several areas, such as history, geography, science, literature, art and sport. These tasks usually require multi-sentence, some of which can be tested at the same time, without strict order. Two typical examples from the dataset include:1. "Among the drafters of the U.S. Declaration of Independence and the French Declaration of the Rights of Man, are there alumni from the same university who were influenced by the Enlightenment?"2. "Did Olympic table tennis champion Zhang Jike and Go world champion Ke Jie both come from coastal provinces?"The ground-truth answers were generated by the LLM and subsequently reviewed and corrected by domain experts.

3.2 Evaluation Metric

To compare the effectiveness of different methods, we adopt accuracy as the evaluation metric, defined as:

$$P=\frac{\#Number\ of\ Correct\ Answers}{\#Total\ Number\ of\ Questions} \tag{3}$$

Only fully correct answers are counted as correct. Answers that are partially correct or contain factual inaccuracies are considered incorrect. However, semantically equivalent responses with different levels of detail are accepted. For example, given the question:"Did Olympic table tennis champion Zhang Jike and Go world champion Ke Jie both come from coastal provinces?"If the standard answer is:"They both come from coastal provinces."Then a model-generated answer like: "Zhang Jike is from Qingdao, Shandong Province, and Ke Jie is from Lishui, Zhejiang Province. They both come from coastal provinces."is considered correct.But an answer like:"Zhang Jike is from Jinan, Shandong Province, and Ke Jie is from Lishui, Zhejiang Province. They both come from coastal provinces." is incorrect, because Zhang Jike is not from Jinan.

3.3 Experimental Results

We compare the following four methods:1) Baseline method that directly submit the question to an LLM for response; 2) Chain-of-Thought (CoT) method [21]; 3) Reflection-based

method [30]; 4) Our proposed method. We evaluate across four different LLMs: Qwen3 (4B), Deepseek-R1 (7B), Gemma3 (4B) and LLaMA 3.2 (3B). The core logic is shared across all models. Due to variation in instruction-following capabilities, we validate that each LLM adheres to expected formats. For instance, if a model fails to return 'Yes', 'No', or 'Uncertain' when prompted, we trigger a reflective correction loop. It should be noticed that since all prompts are in Chinese, and Gemma3 and LLaMA 3.2 are primarily English-language models with relatively small parameter sizes, their performance on some Chinese queries was limited.

The experimental results is shown bellow in table 1.

Table 1: Experimental results of task planing

	Baseline	CoT[21]	RBM[30]	Our method
Qwen3	31	37	45	56
Deepseek r1	25	36	44	53
Gemma3	19	25	23	29
Llama3.2	17	27	25	31

From the table, it can be observed that various methods achieve the best outcomes on the Qwen3 model. This is attributed to Qwen3's superior instruction-following capabilities compared to Deepseek, and under the same parameter conditions, Qwen3 integrates more world knowledge than the Gemma3 4b version. The chain-of-thought method generally outperforms the baseline method, while the reflection method surpasses the chain-of-thought approach. The accuracy of the method proposed in this paper is nearly twice as high as that of the baseline method and also significantly better than both the chain-of-thought method and the reflection method. The granularity of decomposed sub-problems by Qwen3 is typically finer than that of other models. The following table lists the decomposition results of atomic problems for the same question across four different large models.

Table 2: Task decomposition using different LLM

Question	Were there any common alumni from the same university, influenced by the Enlightenment, among the drafters of the American Declaration of Independence and the French Declaration of the Rights of Man and of the Citizen?
Qwen3	1.Who were the drafters of the American Declaration of Independence? 2.Who were the drafters of the French Declaration of the Rights of Man and of the Citizen? 3.Were these drafters all influenced by the Enlightenment? 4.Did these drafters graduate from the same university? 5.Are there any common alumni from the same university, who were influenced by the Enlightenment, among these drafters?
deepseek R1	1.Who were the drafters of the American Declaration of Independence and the French Declaration of the Rights of Man and of the Citizen? 2.What were the educational backgrounds of these drafters? 3.Were the alma maters of these drafters the same university, and is there any alumni relationship between them?
Gemmi 3	1.Who were the drafters of the American Declaration of Independence? 2.Who were the drafters of the French Declaration of the Rights of Man and of the Citizen? 3.Which universities were significantly influenced by the Enlightenment? 4.Were there any university alumni who, during the Enlightenment period, participated in drafting both the Declaration of Independence and the Declaration of the Rights of Man and of the Citizen?
Llama3.2	1.What were the educational backgrounds of the drafters of the American Declaration of Independence? 2.What were the educational backgrounds of the drafters of the French Declaration of the Rights of Man and of the Citizen? 3.Is there a university whose alumni are common to both the drafters of the American Declaration of Independence and the French Declaration of the Rights of Man and of the Citizen?

From the comparison table, it is evident that for the same question, Qwen decomposed it into five subtasks, DeepSeek-R1 into three, Gemmi into four, and LLaMA 3.2 into three. Notably, in the decomposition results of DeepSeek-R1 and LLaMA 3.2, the key information "influenced by the Enlightenment" was omitted, which led to deviation in the final answers.

In the above table above, for example, our method correctly states that the main compelerator of the French Declaration of Human Rights was the Marquis Lafayette, while others were mispined in Robespierre. For example, "Is Japanese scientists going into the invention of the Internet and QR codes? If so, do they work at the same institution?"

4 Conclusions

This article has developed a new task planning and planning structure for intelligent agents, combining a graph based on graphics with large language models (LLM). LLM provides a good understanding of natural language, and graphic structures can be sorted from global dependencies. Collectively, this method has been significantly improved in complex planning scenarios, task accuracy, parallelism and interpretation. Experimental results show that we have a better approach than traditional methods and existing strategies related to LLM in terms of task decomposition, modeling dependency, parallel schedule, and reflection-based optimization. The matrix (EPM) clearly represents a direct and indirect relationship between sub-taches, thus solving the problem of disorders in sequence of execution.

(1) Our method is not entirely based on local ratios in the previous method.

(2) Hibrid pulling-parallel planning mechanism is used. EPM can dynamically monitor the execution of subtaggle, both for the safe and rigorous performance of subtacular tasks, and for the maximum execution of tasks in the logical order, which correctly and maximizes the effectiveness.

(3) The results of the output of the direct binary language model (LLM) are performed to compare the results of the sub-tasting response after the sub-tasting response, which can cause a targeted reflection in the model of a large language.

Our method is much better than the basic method, such as Qwen3, where accuracy increases from 31% to 56%, almost doubles, and was better than the thought chain (CoT) and reflection-based methods. Our method is especially good at distinguishing thin-grained de facto relationships, such as historical functional networks and aligning the entity between domains. Our work provides a new approach to reliable, interpreted by intelligent agents, combining the possibilities of generalization and graphical reasoning.Ivy high-risk applications such as intelligent grid planning, industrial processes management and multi-wheelful dialogue systems. Its modular design allows you to adapt to different areas. In the future, our work will focus on these areas:

(1) Domain-specific Optimization. Combine knowledge of the area and advanced fast engineering to make tasks in areas such as health and financing more accurate disintegration.

(2) Multimodal transportation. The combination of graphical planning and visual time data helps to process more complex inter-midal task reasoning.

(3) Dynamic fitness of the medium.

References

[1] Wang, Y., Pan, Y., Yan, M., Su, Z., & Luan, T. H. (2023). A survey on ChatGPT: AI-generated contents, challenges, and solutions. arXiv preprint. [https://doi.org/10.48550/arXiv.2305.18339]

[2] Enis, M., & Hopkins, M. (2024). From LLM to NMT: Advancing low-resource machine translation with Claude. arXiv preprint. [https://doi.org/10.48550/arXiv.2404.13813]

[3] Anil, R., et al. (2023). Gemini: A family of highly capable multimodal models. arXiv preprint. [https://doi.org/10.48550/arXiv.2312.11805]

[4] Yang, A., et al. (2024). Qwen2 technical report. arXiv preprint. [https://doi.org/10.48550/arXiv.2407.10671]

[5] Bai, X., et al. (2024). DeepSeek LLM: Scaling open-source language models with longtermism. arXiv preprint. https://doi.org/10.48550/arXiv.2401.02954

[6] Xu, H., Kim, Y. J., Sharaf, A., & Awadalla, H. H. (2023). A paradigm shift in machine translation. arXiv preprint. [https://doi.org/10.48550/arXiv.2309.11674]

[7] Huang, X., Liu, W., Chen, X., Wang, X., Wang, H., Lian, D., Wang, Y., Tang, R., & Chen, E. (2024). Understanding the planning of LLM agents: A survey. arXiv preprint. [https://doi.org/10.48550/arXiv.2402.02716]

[8] Zhai, W., Liao, J., Chen, Z., Su, B., & Zhao, X. (2025). A survey of task planning with large language models. Intelligent Computing, 4, Article 0124. [https://doi.org/10.34133/icomputing.0124]

[9] Wang, B., & Deng, Y. (2023). LLM4Agents: A framework for building LLM-powered autonomous agents. arXiv preprint. arXiv:2305.17077

[10] OpenAI. (2023). GPT-4 Technical Report. arXiv preprint. [https://doi.org/10.48550/arXiv.2303.08774]

[11] Ghallab, M., Nau, D., & Traverso, P. (2004). Automated Planning: Theory and Practice. Morgan Kaufmann.

[12] LaValle, S. M. (2006). Planning algorithms. Cambridge University Press.

[13] McDermott, D., et al. (1998). PDDL – The Planning Domain Definition Language. AIPS-98 Planning Competition, 1-27.

[14] Lipovetzky, N., & Geffner, H. (2014). Width and serialization of classical planning problems. Proceedings of the 21st European Conference on Artificial Intelligence (ECAI), 540-545.

[15] Chen, X., et al. (2021). Decision transformer: Reinforcement learning via sequence modeling. Advances in Neural Information Processing Systems (NeurIPS), 34, 15084-15097.

[16] Valmeekam, K., et al. (2023). Large language models still can't plan (well). Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI), 12345-12353.

[17] Liu, X., et al. (2023). LLM+PDDL: Planning with large language models via PDDL translation. Proceedings of the 40th International Conference on Machine Learning (ICML), 9876-9885.

[18] Pan, Y., et al. (2024). Knowledge-aware planning using LLMs. Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI), 5432-5440.

[19] Singh, S., et al. (2023). ProgPrompt: Generating programs with planning in the loop. Proceedings of the 11th International Conference on Learning Representations (ICLR).

[20] Yao, S., et al. (2022). ReAct: Synergizing reasoning and acting in language models. Advances in Neural Information Processing Systems (NeurIPS), 35, 22199-22213.

[21] Wei, J., et al. (2022). Chain of thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems (NeurIPS), 35, 24824-24837.

[22] Shen, Y., et al. (2023). HuggingGPT: Solving AI tasks with ChatGPT and its friends in HuggingFace. Proceedings of the 40th International Conference on Machine Learning (ICML), 3210-3219.

[23] Gao, Q., et al. (2023). Plan-and-solve prompting. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 1234-1245.

[24] Wang, T., et al. (2022). Self-consistency improves chain of thought reasoning in LLMs. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL), 654-665.

[25] Yao, S., et al. (2023). Tree of thoughts: Deliberate problem solving with large language models. Proceedings of the 40th International Conference on Machine Learning (ICML), 4321-4330.

[26] Zhao, W., et al. (2023). LLM-MCTS: Monte Carlo tree search with large language models. Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI), 7654-7662.

[27] Xiao, Z., & Wang, H. (2023). LLM-A: Solving problems with A search augmented large language models. Proceedings of the 40th International Conference on Machine Learning (ICML), 8765-8774.

[28] Dagan, A., et al. (2023). LLM-DP: Planning in dynamic environments with LLMs and PDDL. Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI), 5432-5440.

[29] Guan, L., et al. (2023). Planning with language models via heuristic PDDL translation. Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI), 9876-9885.

[30] Lin, Z., et al. (2023). SwiftSage: Fast and reliable tool-augmented LLM planning. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1234-1245.

[31] Shinn, N., et al. (2023). Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems (NeurIPS), 36, 4321-4330.

[32] Madaan, A., et al. (2023). Self-refine: Iterative refinement with self-feedback. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 654-665.

[33] Gou, Z., et al. (2023). CRITIC: Language agents with self-correcting tool use. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 3210-3219.

[34] Park, J., et al. (2023). Generative agents: Interactive simulacra of human behavior. Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST), 1-22.

[35] Zhong, Y., et al. (2023). MemoryBank: Enhancing LLMs with long-term memory. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 7654-7662.

[36] An, J., et al. (2023). LEMA: Large language model-enhanced multi-agent planning. Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI), 4321-4330.

[37] Chen, J., et al. (2022). Program of Thought: Reasoning via code generation. Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS), 15084-15097.

[38] Zhang, J., et al. (2023). AgentTuning: Enhancing LLMs via agent-aligned instruction tuning. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 9876-9885.

[39] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., & Manning, C. D. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2369-2380.