Fuzzy Systems and Data Mining X
A.J. Tallón-Ballesteros (Ed.)

doi:10.3233/FAIA241464

LLM-AS: A Self-Improve LLM Reasoning Framework Integrated with A* Heuristics Algorithm

Xueqi MENG [a], Kun NIU [a,1] and Xiao CHEN [a]
[a] *School of Computer Science, Beijing University of Posts and Telecommunications, China*
ORCiD ID: Xueqi Meng https://orcid.org/0009-0006-1003-3684, Kun Niu https://orcid.org/0000-0003-1877-5982, Xiao Chen https://orcid.org/0009-0008-2181-0084

Abstract. To address the limitations of flexibility or efficiency of existing prompting paradigms in generating intermediate reasoning steps, this paper proposes an reasoning framework LLM-AS, which innovatively combines the A* search algorithm with the reasoning process of large language models(LLMs). LLM-AS utilizes the efficient exploration capability of the A* algorithm and avoids the redundant exploration of high-cost nodes, which significantly improves the search efficiency and reduces the cost of invoking LLM. Meanwhile, through the self-improve mechanism of LLMs, LLM-AS ensures the quality of the generated solutions while minimizing model interactions. In addition, the flexibility of the A* search algorithm enables LLM-AS to be applicable to diverse thought organization structures, providing more possibilities for handling various tasks. We conducted experiments on two complex tasks, game 24 and 8 puzzle, to compare the accuracy of the existing prompting paradigms and LLM-AS on both gpt-3.5-turbo and gpt-4.0. The experimental results show that LLM-AS effectively improves the ability of LLMs to solve complex tasks.

Keywords. large language models, A* search algorithm, reasoning and planning

1. Introduction

Recent advances in large language models (LLMs) have opened up more possibilities for solving different types of problems in natural language processing [1–6]. Standard I/O prompts work for simple, single-step problems. CoT [7] and CoT-SC [8] improve performance by generating intermediate steps but are limited to linear thought. ToT [9] organizes thoughts in a tree structure, improving performance on complex tasks but increasing invocation costs. The question arises: is it possible to reduce the cost of invocation while guaranteeing the performance of LLMs in solving complex problems?

We propose LLM-AS, a reasoning framework for complex tasks that combines A* search [10] with LLM reasoning. It enhances exploration efficiency by avoiding high-

[1] Corresponding Author: Kun Niu, niukun@bupt.edu.cn.

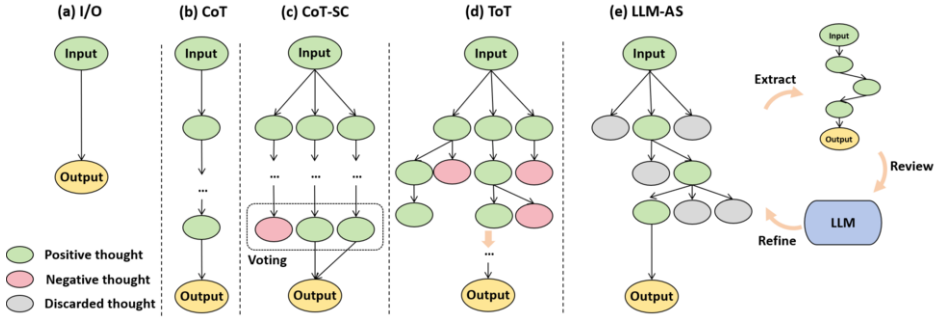

Figure 1. Comparison of our LLM-AS with other prompting paradigms.

cost nodes and exploring towards the target, reducing LLM invocation costs while improving efficiency.

To minimize A* search risks, we use LLMs for self-review and correction. The self-improve module optimizes solutions iteratively, enhancing quality of the generated solutions while minimizing LLM interactions.

Incorporating the A* search algorithm offers flexibility as it can explore various thought structures and allows intelligent heuristic design, adapting the reasoning framework to more task types.

We conducted experiments with LLM-AS using gpt-3.5-turbo [11] and gpt-4.0 [12] on two complex real-world tasks, Game 24 and 8 Puzzle, which shows that our LLM-AS exhibits superior performance. It is an effective thoughts generation method and provides new insights for solving complex problems with LLMs.

Our main contributions are as follows:

- We propose a new planning and reasoning algorithm by combining LLMs and A* search, optimizing components for enhanced LLM reasoning.
- We propose a self-improve module to balance LLM invocation cost and solution accuracy by iteratively optimizing solutions from a reasoning process integrating A* search.
- We experimentally prove our reasoning algorithm's effectiveness on complex tasks, improving LLM performance for solving complex problems.

2. Background

2.1. Reasoning for LLMs

Prompts can enhance LLMs' reasoning by guiding them to break down complex problems into intermediate steps, mimicking human thinking [13, 14]. Each step's outcome can be organized in various structures [7, 9, 15–17], like linear chains or hierarchical trees (Fig. 1), related to the reasoning approach.

Input-Output (IO) Prompting. The I/O method is the simplest way to utilize LLMs for task resolution. It only requires defining the task and problem as a prompt input to the model, with the model's output representing the solution.

Chain-of-Thought [7]. The CoT method leverages the context-learning abilities of LLMs by breaking a problem into multiple intermediate steps. It provides the model with a prompt in the form of a triplet ¡input, intermediate steps, output¿, thereby guiding the model through the problem-solving process.

Self-consistency CoT (CoT-SC) [8]. The CoT-SC method samples different reasoning processes to generate multiple solutions for a given problem and selects the most consistent answer as the final solution from the model.

Tree-of-Thought (ToT) [9]. The ToT method organizes the thought nodes for problem-solving in a tree structure. ToT allows for backtracking and foresight when necessary to explore globally optimal solutions.

2.2. A Search Algorithm*

The A* search algorithm (1968) [10] is an efficient heuristic method using an evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the actual cost to the current node and $h(n)$ estimates the cost to the target. This prioritizes exploring paths closer to the goal, reducing unnecessary search space exploration and enhancing efficiency.

$$f(x) = g(x) + h(x) \tag{1}$$

The A* algorithm is often used to solve path planning problems. Currently, there are also efforts to apply LLMs to the traditional A* algorithm to address computational and memory efficiency issues in path planning. [18]

2.3. LLMs self-improve

When dealing with complex and tedious tasks, human beings often complete a simple initial solution and then gradually make adjustments and refinements based on it. LLMs can also improve the accuracy of their solutions by simulating such a way of thinking and conducting self-reflection [19, 20]. The self-reflection capability of LLMs refers to the ability of the model to review, evaluate and improve its own output after generating it. This capability of LLMs not only further improves the quality of generation and factual accuracy, but also maintains the original creativity of LLMs [21–23]. This ability is often applied to improve the decision-making ability of agents [24]. And, some methods that can enhance the ability of LLMs to perform self-reflection are also currently proposed [25] and can also be used as our future work to optimize the components in our framework.

3. Methodology

We propose an inference strategy combining LLMs with the A* search to enhance LLMs' reasoning on complex problems, as shown in Fig. 2. In the following, we introduce its components and workflow.

Initialization. In initialization, we define states (S), the set of executable actions (A), and the state transfer function (δ: $S \times A \rightarrow S$). The initial state (s_0) is added to the open list for unexplored nodes, while the closed list for accessed nodes is initialized as empty.

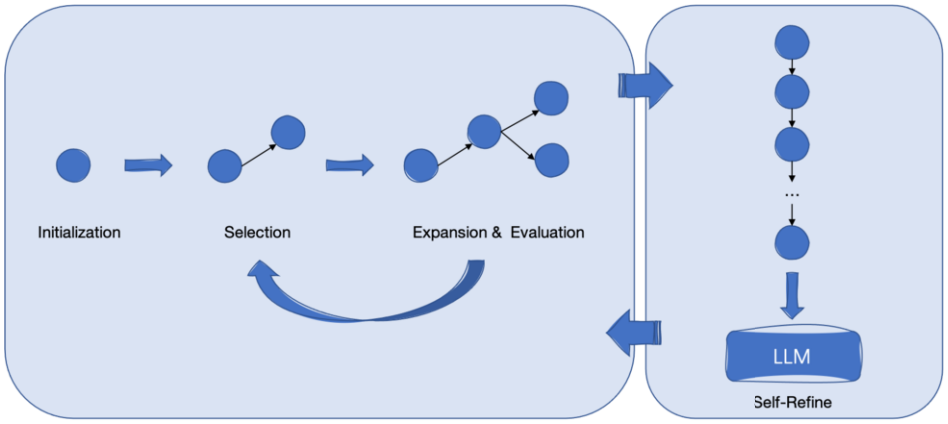

Figure 2. The components and workflow of our LLM-AS framework.

Selection. The node with the lowest cost $f(s) = g(s) + h(s)$ in the open list is chosen, moved to the closed list. $g(s)$ is the actual cost from the initial state to the current state. $h(s)$ estimates the cost to the goal (lower means closer).

Expansion and Evaluation. Successor states are extended based on action spaces and checked against the closed list to avoid loops. Their $g(s)$ and $h(s)$ values are updated based on the state transition costs. An accurate $h(s)$ optimizes A* search performance. For tasks which determining the heuristic function is challenging, we use LLMs to compute $h(s)$ using their internal knowledge.

Termination Conditions. The iterative process of selection, expansion, and evaluation continues until termination condition T is met, improving answers and exploring new possibilities. T can be set as follows:

- Reaching the goal state: For tasks with a clearly defined goal, this condition directly triggers the end of the search when the search reaches a preset goal state.
- Search constraints: To control computational resource consumption, limits on search depth or the number of nodes visited can be set.

Review and Correction. After generating a solution through the reasoning process of LLMs integrated with the A* search algorithm, the self-improve module is invoked. Initially, the path of thoughts is organized into structured input prompts according to task execution steps, converting it into a text format understandable by the LLM. converted into a text format understandable by the large model. Subsequently, the LLM conducts a thorough review of the generated solution to identify potential logical errors or inconsistencies [21–23]. Based on the prompts, if no errors are identified, the solution is deemed final. Otherwise, the LLM outputs the steps where errors occur. Based on the feedback from the model, iterative corrections are made to the preceding reasoning process to continuously optimize the solution's quality. Starting from the step where an error is identified, subsequent steps are regenerated. This module exemplifies the complementary advantages of the LLM's self-improve capabilities and the reasoning integrated with the A* search algorithm, jointly driving the effective and efficient resolution of complex problems.

Table 1. An overview of definitions for tasks

	game 24	**8 puzzle**
Thought	intermediate equations	swap and the puzzle state after the move
State	the remaining 1-4 numbers	the number layout in the grid
Action	picking two number and a operation to compose an equation	swap blank square with one of its neighboring numbered squares
initial state	four numbers that can be used to form a mathematical expression	a grid with 8 number squares and 1 blank square
goal state	the result of the math of the expression composed with the remaining numbers is 24	The numbers are arranged in the order of 1-8 from the top left to the bottom right, with the blank square at the bottom right corner

4. Experiment

We evaluated the effectiveness of the LLM-AS approach on two challenging tasks: game 24 and 8 puzzle. These tasks require multiple steps to complete. The high complexity and difficulty of these tasks pose significant demands on the reasoning capabilities of LLMs. To demonstrate the effectiveness of LLM-AS, we compared it against standard I/O, zero-shot CoT, few-shot CoT, CoT-SC, and ToT. During the experiments, we set the temperature of the invoked LLM to 0.0.

4.1. game 24

Game 24 is a mathematical reasoning task where the goal is to form an arithmetic expression using the given 4 numbers and basic operations (+, -, *, /) so that the result is 24.

4.1.1. Task Setup

Our dataset has 1362 games from 4nums.com, with 100 randomly selected for testing. The adaptability definition for the game 24 task is outlined in Table 1, where thoughts are equations and states are remaining numbers. The goal is for the equation to result in 24. We used the success rate across 100 games as the evaluation metric.

4.1.2. Baseline & LLM-AS Setup

We provide five examples for the standard I/O prompt. For few-shot CoT, we use intermediate equations. For CoT-SC, we sample 10 times and voted for the most consistent result. For ToT, the model generate and evaluate next steps. We explore candidate set widths b=1 and b=3 for comparison. For LLM-AS, the LLM generate and evaluate states. Five examples are given for evaluation. After A* search, the LLM review and correct the solution using five refine examples. In the refine prompt, each time a number is selected, it is regarded as a step and labeled sequentially to allow the LLM to identify the specific step where the error occurs.

4.1.3. Results

Table 2 shows each method's performance on game 24 with gpt-3.5-turbo and gpt-4.0. LLM-AS consistently outperforms baselines, with improved accuracy to 21.56% on gpt-3.5-turbo and 65.32% on gpt-4.0 after one correction. This highlights LLM-AS's superiority in complex tasks. The best baseline, ToT (b=3) on gpt-4.0, achieves 62.00% accuracy but costs more due to many LLM invocations. LLM-AS improves accuracy while reducing explorations and costs.

4.2. 8 puzzle

8 puzzle is a classic puzzle game. The game panel is a 3x3 grid with a number (1-8) or a blank square (represented by 0) on each small square. The goal of the game is to arrange the numbers in the order 1-8 from the top left to the bottom right by exchanging the blank squares with the neighboring number squares, with the blank square in the bottom right corner.

4.2.1. Task Setup

8 Puzzle is a classic game with a 3x3 grid of numbers 1-8 and a blank (0). The goal is to arrange numbers 1-8 in order by swapping them with the blank, ending with the blank in the bottom right. We used the proportion of these 100 games reaching the goal state as our evaluation metric.

4.2.2. Baseline & LLM-AS Setup

We provide three examples for standard I/O. In few-shot CoT, we concatenate actions with the state. For CoT-SC, we sample 10 times and chose the most consistent result. ToT generate and evaluate next steps, with a max search depth of 9. We try candidate set widths b=1 and b=3. For LLM-AS, the model evaluate states using three examples, then review and correct the solution using three more examples after A* search. In the refine prompt, each swap action is numbered sequentially enabling the LLM to pinpoint the exact step where the error arises.

4.2.3. Results

The spatial complexity and long-term planning requirements of the 8 puzzle significantly challenge LLMs. Table 2 shows LLM-AS outperforms all baselines, improving to 30.63% and 39.92% on gpt-3.5-turbo and gpt-4.0 after one correction. This surpasses the best baseline, ToT (b=3), on gpt-4.0 with 15.00% accuracy. LLM-AS's integration of A* search and LLM-based evaluation effectively tackles the puzzle, showing potential for spatial reasoning and long-term planning tasks.

As observed in Table 2, both the baseline methods and our LLM-AS exhibit superior performance on gpt-4.0 compared to gpt-3.5-turbo. gpt-4.0 boasts a larger model size and a more extensive and diverse training dataset than gpt-3.5-turbo. Furthermore, gpt-4.0 ingeniously leverages Graph Neural Networks, enabling the model to better capture the temporal information and dependencies in language. We attribute these factors to the performance disparity between gpt-3.5-turbo and gpt-4.0, resulting in gpt-4.0's outstanding performance in handling the two complex tasks.

Table 2. Experimental results on game 24 and 8 puzzle tasks

	game 24		8puzzle	
	gpt-3.5-turbo	gpt-4.0	gpt-3.5-turbo	gpt-4.0
I/O	5.98	8.90	0.00	1.34
zero-shot CoT	5.98	8.92	0.00	1.92
few-shot CoT	3.18	4.05	0.00	5.04
CoT-SC	3.20	4.38	3.78	6.94
ToT(b=1)	8.02	40.13	7.86	8.01
ToT(b=3)	15.71	62.00	13.26	15.00
LLM-AS	21.56	65.32	30.63	39.92

5. Conclusion

This paper shows LLM-AS enhances LLMs' reasoning for complex tasks by integrating A* search and utilizing the LLMs for review and correction, improving accuracy and reliability. A* search prioritizes goal-directed paths, reducing high-cost exploration, significantly enhancing efficiency and lowering LLM invocation costs. Experiments on two tasks show LLM-AS outperforms existing prompt paradigms.

This research advances the application of LLMs in complex tasks, laying a foundation for enhancing their planning and decision-making capabilities in agents [26,27]. For future work, we will conduct more experiments to explore the capability of LLM-AS in exploring more diverse thought topologies, such as graph structures. And apply it to a wider range of real-world problems.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020, https://doi.org/10.48550/arXiv.2005.14165.

[2] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[4] R. Omar, O. Mangukiya, P. Kalnis, and E. Mansour, "Chatgpt versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots," *arXiv preprint arXiv:2302.06466*, 2023, https://doi.org/10.48550/arXiv.2302.06466.

[5] S. Frieder, L. Pinchetti, R.-R. Griffiths, T. Salvatori, T. Lukasiewicz, P. Petersen, and J. Berner, "Mathematical capabilities of chatgpt," *Advances in neural information processing systems*, vol. 36, 2024, https://doi.org/10.48550/arXiv.2301.13867.

[6] G. Kim, P. Baldi, and S. McAleer, "Language models can solve computer tasks," *Advances in Neural Information Processing Systems*, vol. 36, 2024, https://doi.org/10.48550/arXiv.2303.17491.

[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022, https://doi.org/10.48550/arXiv.2201.11903.

[8] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022, https://doi.org/10.48550/arXiv.2203.11171.

[9] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024, https://doi.org/10.48550/arXiv.2305.10601.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968, 10.1109/TSSC.1968.300136.

[11] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022, https://doi.org/10.48550/arXiv.2203.02155.

[12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[13] A. Newell, J. C. Shaw, and H. A. Simon, "Report on a general problem solving program," in *IFIP congress*, vol. 256. Pittsburgh, PA, 1959, p. 64.

[14] H. A. Simon and A. Newell, "Human problem solving: The state of the theory in 1970." *American psychologist*, vol. 26, no. 2, p. 145, 1971, https://doi.org/10.1037/h0030806.

[15] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le *et al.*, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022, https://doi.org/10.48550/arXiv.2205.10625.

[16] A. Creswell and M. Shanahan, "Faithful reasoning using large language models," *arXiv preprint arXiv:2208.14271*, 2022, https://doi.org/10.48550/arXiv.2208.14271.

[17] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 682–17 690, https://doi.org/10.1609/aaai.v38i16.29720.

[18] S. Meng, Y. Wang, C.-F. Yang, N. Peng, and K.-W. Chang, "Llm-a*: Large language model enhanced incremental heuristic search on path planning," 2024, https://doi.org/10.48550/arXiv.2407.02511. [Online]. Available: https://arxiv.org/abs/2407.02511

[19] W. Saunders, C. Yeh, J. Wu, S. Bills, L. Ouyang, J. Ward, and J. Leike, "Self-critiquing models for assisting human evaluators," 2022, https://doi.org/10.48550/arXiv.2206.05802. [Online]. Available: https://arxiv.org/abs/2206.05802

[20] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen, and J. Gao, "Check your facts and try again: Improving large language models with external knowledge and automated feedback," 2023, https://doi.org/10.48550/arXiv.2302.12813. [Online]. Available: https://arxiv.org/abs/2302.12813

[21] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang *et al.*, "Self-refine: Iterative refinement with self-feedback," *Advances in Neural Information Processing Systems*, vol. 36, 2024, https://doi.org/10.48550/arXiv.2303.17651.

[22] D. Paul, M. Ismayilzada, M. Peyrard, B. Borges, A. Bosselut, R. West, and B. Faltings, "Refiner: Reasoning feedback on intermediate representations," *arXiv preprint arXiv:2304.01904*, 2023, https://doi.org/10.48550/arXiv.2304.01904.

[23] Y. Xie, K. Kawaguchi, Y. Zhao, J. X. Zhao, M.-Y. Kan, J. He, and M. Xie, "Self-evaluation guided beam search for reasoning," *Advances in Neural Information Processing Systems*, vol. 36, 2024, https://doi.org/10.48550/arXiv.2305.00633.

[24] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024, https://doi.org/10.48550/arXiv.2303.11366.

[25] W. Zhang, Y. Shen, L. Wu, Q. Peng, J. Wang, Y. Zhuang, and W. Lu, "Self-contrast: Better reflection through inconsistent solving perspectives," *arXiv preprint arXiv:2401.02009*, 2024, https://doi.org/10.48550/arXiv.2401.02009.

[26] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents," *arXiv preprint arXiv:2302.01560*, 2023, https://doi.org/10.48550/arXiv.2302.01560.

[27] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530, https://doi.org/10.48550/arXiv.2209.11302.