LLM-based Reward Engineering for Reinforcement Learning: A Chain of Thought Approach

1st Xinning Zhu
Sino-European School of Technology
Shanghai University
China
zhuxinning@shu.edu.cn

2nd Jinxin Du*
Sino-European School of Technology
Shanghai University
China
jinxin_du@shu.edu.cn
*Corresponding author

3rd Qiongying Fu*
Sino-European School of Technology
Shanghai University
China
fqiongying@163.com
*Corresponding author

4th Lunde Chen*
Sino-European School of Technology
Shanghai University
China
lundechen@shu.edu.cn
*Corresponding author

***Abstract*—Reinforcement Learning (RL) has achieved significant milestones in various domains. Traditional reward engineering often requires extensive domain knowledge and trial-and-error experimentation. This paper explores the integration of Large Language Models (LLMs) with CoT reasoning to automate and enhance the generation of reward functions for RL environments. We present a comprehensive methodology that leverages CoT-enabled LLMs to generate sophisticated, adaptive, and context-aware reward functions, implemented within Gymnasium environments and trained using Stable Baselines3. Through a detailed case study on the Bipedal Walker environment, we demonstrate the efficacy of our approach in producing superior agent performance compared to non-CoT approaches.**

***Index Terms*—Reinforcement Learning, Large Language Models, Chain of Thought, Reward Engineering, Gymnasium**

I. Introduction

The success of Reinforcement Learning (RL) agents is heavily dependent on the design of reward functions that accurately reflect desired behaviors and objectives [1]. Traditionally, reward engineering has been a manual and iterative process, often requiring deep expertise in the specific environment and task [2]. As RL applications become more complex, the need for automated and intelligent reward function generation becomes paramount.

Large Language Models (LLMs) have shown remarkable capabilities in understanding and generating human-like text, which can be harnessed for various applications beyond natural language processing [3]. One such application is in the domain of RL reward engineering. By utilizing LLMs with Chain of Thought (CoT) reasoning, it is possible to generate reward functions that are not only contextually relevant but also logically structured, reducing the reliance on manual intervention [4].

This paper investigates the use of CoT-enabled LLMs for generating reward functions in RL environments. The process begins with an initial system prompt that provides a base description of the task and the environment (Fig. 1). This prompt includes coding instructions and environment code, which serve as the foundation for the LLM to understand the context and requirements of the task [5].

The CoT integration involves a series of steps where the LLM analyzes the task description and environment code carefully. It identifies key objectives and constraints, breaks down how different actions should be rewarded, and considers potential edge cases. This structured reasoning process ensures that the generated reward function is both contextually relevant and logically sound [6].

The LLM then iteratively refines the reward function through multiple iterations, each guided by the CoT analysis instructions. These instructions include performance analysis, reward function analysis, behavioral analysis, and improvement planning. The best sample from all iterations is selected and used to train the RL agent in the customized environment.

We focus on the Gymnasium framework and employ Stable Baselines3 for training agents [7]. Through a case study on the Bipedal-Walker environment, we illustrate the process and effectiveness of our approach. The results demonstrate that CoT-enabled LLMs can significantly improve the efficiency and quality of reward function generation, leading to better-performing RL agents.

In summary, this study highlights the potential of integrating CoT reasoning into LLMs for automating the reward engineering process in RL. By leveraging the structured and logical capabilities of CoT, we aim to reduce the manual effort required and enhance the robustness and reliability of RL applications.

II. Related Work

A. Reward Function Design

RL heavily depends on the design of reward functions, which are notoriously challenging to construct. Traditional

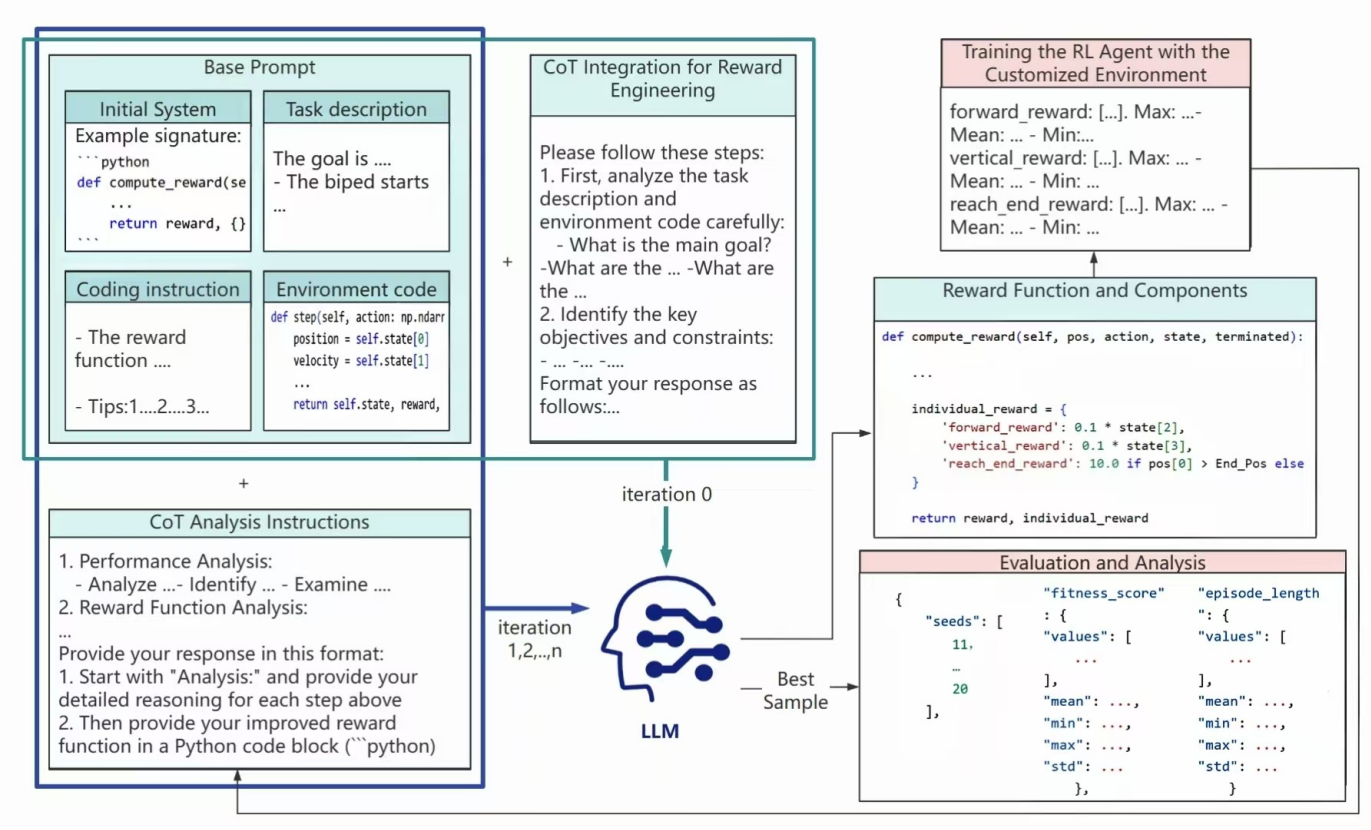

Fig. 1. Procedure of the Reward Engineering Framework with CoT Integration.

methods have relied on domain-specific knowledge and manual crafting, where experts define rewards based on task characteristics. However, as tasks grow in complexity, these methods become impractical, leading to either sparse or dense rewards that lack intermediate guidance or risk overfitting to local optima [8].

Advanced techniques such as Reward Shaping introduce auxiliary rewards to facilitate faster convergence without altering the optimal policy [9]. Potential-Based Reward Shaping (PBRS), proposed by Ng et al., ensures policy invariance under certain transformations [10]. Inverse Reinforcement Learning (IRL) aims to infer reward functions from expert demonstrations, reducing human effort in reward design [11]. Notable IRL methods include Maximum Entropy IRL [12] and Guided Cost Learning [13].

Intrinsic motivation methods, like curiosity-driven learning, promote exploration by rewarding agents for visiting novel states or acquiring new skills, addressing issues associated with sparse rewards [14] [15]. These methods enhance the agent's ability to explore its environment effectively, particularly when the reward signal is sparse or poorly defined.

Recent advancements have seen the integration of LLMs into the process of reward function design for RL. This approach leverages the natural language understanding capabilities of LLMs to automate the creation of reward functions, offering a promising alternative to traditional methods. Kwon et al. [16] utilized GPT-3 as a proxy reward function, demonstrating superior performance over supervised learning-based rewards even in the absence of examples. Xie et al. [17] introduced the TEXT2REWARD framework, which automatically generates dense reward functions from human feedback and achieves comparable results to manually crafted rewards in robotic manipulation tasks. Ma et al. [18] developed the EUREKA framework, combining environmental context, evolutionary search, and reward reflection to produce effective reward functions, especially for complex tasks. Song et al. [19] proposed a self-refined LLM framework that designs initial reward functions based on natural language input and iteratively refines them to align with task requirements.

These studies highlight the potential of LLMs in enhancing RL by automating reward design, improving system interpretability, and expanding applicability to more complex and varied tasks. The integration of LLMs with RL opens up new possibilities for solving intricate real-world problems while maintaining high levels of efficiency and reliability.

B. Chain of Thought in LLMs

Chain-of-Thought (CoT) prompting enhances reasoning capabilities in LLMs, addressing limitations in multi-step reasoning tasks despite improvements in model size and training data [20]. CoT leverages few-shot learning, where models

learn from examples containing intermediate reasoning steps, promoting structured thinking patterns.

Few-shot CoT uses limited demonstrations that include both problem statements and corresponding chains of thought, aiding models in breaking down complex problems [21]. Self-consistency generates multiple reasoning paths and selects the most consistent answer through voting mechanisms, improving robustness and accuracy [22]. Verifier-based approaches train a component to assess the correctness of generated reasoning paths, ensuring higher quality outputs [23].

Empirical studies demonstrate significant performance improvements in reasoning tasks such as arithmetic and commonsense reasoning when applying CoT, particularly on benchmarks like MultiArith and GSM8K [24]. CoT also increases the transparency and interpretability of LLM-generated outputs, facilitating broader applications in sophisticated reasoning domains [25].

III. Methodology

In this section, we outline the methodology employed in our research to integrate CoT reasoning into the generation of reward functions for RL agents.

A. Prompt Design for CoT-Enabled Reward Generation

The initial phase involves using CoT reasoning, guided by a LLM, to generate structured reward function components $r_i(s, a)$, which are dependent on states s and actions a. Each component originates from specific input information I_i:

$$r_i(s, a) = \text{CoT}(I_i) \tag{1}$$

Simultaneously, LLM assigns an initial weight w_i to each component, establishing the base reward function:

$$R_{base} = \sum_{i=1}^{m} w_i \cdot r_i(s, a) \tag{2}$$

This ensures that each reward function component is underpinned by coherent thought processes and logically contributes to achieving the task objectives.

B. Leveraging CoT in Reward Function Generation

The RL agent is trained within the customized Gymnasium environment using Stable Baselines3 (SB3). The training process follows the policy gradient update rule, incorporating the CoT-generated reward function to guide the learning process:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta; \mathbf{s}, \mathbf{a}) \tag{3}$$

where θ denotes the policy parameters, α is the learning rate, and $J(\theta; \mathbf{s}, \mathbf{a})$ represents the objective function over states $\mathbf{s}$ and actions $\mathbf{a}$.

The performance of each reward function component during training is evaluated using fitness scores $FS(r_i^{(k)})$. The CoT reasoning process extracts logical reasoning and structured reward components from the LLM's response, which can then be iteratively improved based on observed performance metrics and domain knowledge. Subsequently, through CoT reflection, adjustments are made to the reward function components and their weights for the next iteration:

$$R_{new}^{(k)} = \text{CoT\_Reflection}(FS(r_i^{(k)}), W^{(k)}) \tag{4}$$

where $W^{(k)}$ represents the set of weights at iteration k.

C. Final Evaluation and Analysis

Throughout these iterations, the best-performing reward function configuration and its corresponding weights are tracked and updated:

$$\text{best\_config}^{(k)} = \arg\max_{j \le k} \{FS(r_i^{(j)}, W^{(j)})\} \tag{5}$$

Upon completing all n iterations, a final evaluation is performed on the overall best sample identified across all iterations. This assessment involves re-evaluating the top-performing sample with different random seeds to ensure robustness and generalizability. By doing so, we gain deeper insights into the stability and reliability of the reward function and its components, ensuring that the selected sample can perform well under varying conditions.

D. Summary

By continuously evaluating and adjusting the components and weights of reward functions using CoT reasoning, this method enhances the precision with which reward functions reflect task goals. It promotes more efficient learning and improved performance of RL agents. The iterative refinement process, combined with final evaluation, ensures the optimization of reward function configurations while maintaining clarity and effectiveness throughout the training process. CoT not only guides the generation and adjustment of reward functions but also ensures that the reasoning behind these adjustments remains transparent and logical, thereby enhancing the overall quality of the RL agent's training.

IV. Experimental Procedure

This section details the implementation of our methodology in the Bipedal Walker environment, following the theoretical framework established in Section III. We present the experimental protocol that demonstrates how CoT reasoning guides the iterative improvement of reward functions. The full appendix is available at GitHub[1].

A. Experimental Setup and Configuration

The experiments were conducted using Gymnasium's BipedalWalker-v3 environment, with the PPO algorithm implemented through Stable-Baselines3 serving as the primary training method. Each training iteration consisted of 1 million steps to ensure sufficient learning opportunity. The computational infrastructure comprised two NVIDIA GeForce RTX 3090 GPUs with 24GB memory each, while Llama-3.1 was employed for CoT reasoning processes.

[1]https://github.com/evidentiallab/RewardEngineering_CoT/blob/main/appendix.pdf

To evaluate the performance, we established multiple assessment criteria aligned with the theoretical framework presented in Section III. The primary evaluation metrics include the fitness score reflecting the agent's performance, episode length indicating task completion efficiency.

B. CoT-Guided Reward Function Design

The reward function design process follows three main phases, implementing the theoretical framework developed in Section III. The process begins with an initial design phase, followed by iterative refinement, and concludes with environment integration.

1) Initial CoT-Based Design Phase: The initial phase commences with a task analysis, where we decompose the Bipedal Walker objectives into fundamental components. Following (1), we derive the initial reward components $r_i(s, a)$ through CoT reasoning, as shown in Fig. 2. These components are then assigned preliminary weights according to (2), establishing the foundation for subsequent refinement.

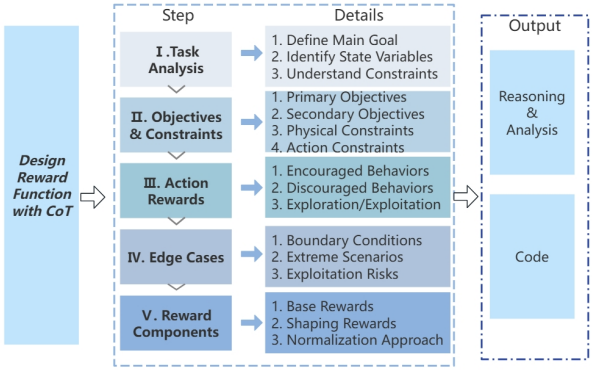

Fig. 2. CoT-Based Reward Function Design Process.

The refinement process implements (3), encompassing a systematic analysis of agent behavior and performance metrics.

2) CoT-Guided Iterative Refinement Phase: The refinement process implements (4), encompassing a systematic analysis of agent behavior and performance metrics, as shown in Fig. 3. Through this analysis, we continuously adjust both the reward components $r_i(s, a)$ and their associated weights W_i. This iterative process ensures that the reward function evolves to better align with desired behaviors.

C. Training and Evaluation Protocol

The evaluation phase involves integrating the refined reward function into the Gymnasium environment. This step requires implementing a custom environment wrapper to apply the new reward function, along with normalization and scaling procedures. The modified environment is then integrated into the PPO training pipeline to ensure consistent and reliable training processes.

We begin by establishing a benchmark through training the agent using the default reward function and recording

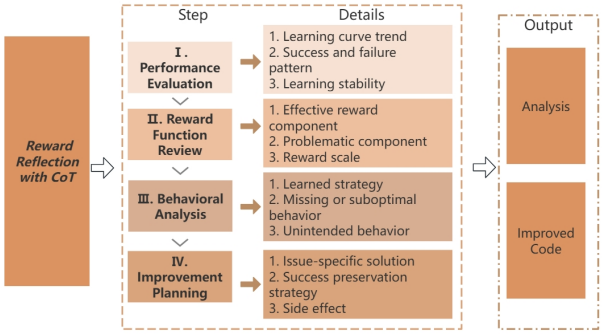

Fig. 3. CoT-Based Reward Function Reflection and Refinement Process.

performance metrics. This baseline serves as a reference for subsequent comparisons.

In iterations 0 through n, we implement the CoT-based refined reward function. Each iteration generates a new reward function variant used to train the agent. Performance data across all relevant metrics are collected during these runs.

Once the best-performing sample is identified from the iterative process, it undergoes re-evaluation with different random seeds. This ensures that the robustness of the reward function is assessed across various initial conditions, providing a more comprehensive understanding of its stability and reliability.

The outcomes of this experimental procedure are presented in Section V, where we provide an in-depth analysis of our CoT-based approach's effectiveness.

V. Results of Experiment

In this section, we present the results obtained from applying the proposed methodology to the Bipedal Walker environment. The focus is on demonstrating the outcomes of using CoT reasoning for reward engineering and comparing these outcomes against existing benchmarks. Additionally, ablation studies provide insights into the contribution of individual components.

A. Using CoT to start from scratch

The experiment utilizing CoT reasoning for designing a reward function from scratch in the Bipedal Walker environment has yielded significant improvements in the agent's performance metrics. The fitness score, serving as an indicator of the agent's proficiency within the environment, was meticulously tracked over multiple iterations.

As shown in Fig. 4, the fitness score rapidly increased during the early iterations.

The experimental results demonstrate that the CoT-based reward function effectively guided the agent through its learning process. Starting from an initial state where the agent exhibited no proficient behavior, the fitness score rapidly increased during the early iterations. This sharp rise is indicative of the agent acquiring essential skills necessary for navigating the Bipedal Walker environment successfully. Subsequently, the fitness score reached a plateau at approximately 300 points,

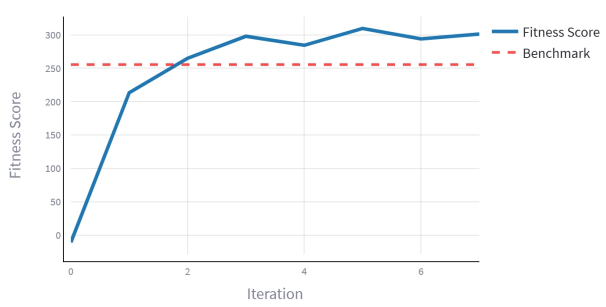

Fig. 4. Fitness Score Per Episode.

reflecting the stabilization of the agent's performance at a high level of proficiency.

In summary, the application of CoT reasoning in this experiment resulted in a reward function that improved the agent's performance in the Bipedal Walker environment. The fitness score's evolution over iterations provides evidence of the method's effectiveness in guiding the agent towards optimal behaviors, achieving both rapid learning and stable performance.

B. Comparison with Benchmark

The experiment comparing the performance of the CoT-based reward function against benchmark methods has yielded insightful results, as shown in Table I and II. The metrics evaluated include fitness score, reward, and episode length. These metrics provide a comprehensive view of the agent's performance in the Bipedal Walker environment.

TABLE I
PERFORMANCE METRICS COMPARISON FOR BATCH SIZE = 32

Category	**Metric**	***CoT-Based***	***Benchmark***
Fitness Score	Mean	274.1469	245.3074
	Std	82.5062	105.4078
	Max	304.2816	298.8206
	Min	-106.2407	-109.3440
	Smoothness	6.8830	38.9018
Episode Length	Mean	1060.0550	1068.2400
	Std	183.9346	286.2089
	Max	1321.5000	1600.0000
	Min	123.8750	47.1250
	Smoothness	33.6553	140.8308

TABLE II
PERFORMANCE METRICS COMPARISON FOR BATCH SIZE = 64

Category	**Metric**	***CoT-Based***	***Benchmark***
Fitness Score	Mean	248.4897	222.0471
	Std	83.0830	115.2739
	Max	301.2007	295.9536
	Min	-94.2776	-121.4410
	Smoothness	21.6980	47.4225
Episode Length	Mean	1016.5600	1186.0525
	Std	186.2482	330.2637
	Max	1178.0000	1600.0000
	Min	95.8750	40.0000
	Smoothness	74.9583	177.3598

The comparison between the sample metrics (using the CoT-based reward function) and the benchmark metrics reveals several key insights:

- Fitness Score: The mean fitness score for the CoT-based reward function is higher compared to the benchmark, indicating better overall performance.
- Episode Length: The mean episode length for the CoT-based reward function is slightly lower compared to the benchmark, suggesting that the agent using the CoT-based reward function completes tasks more efficiently.
- Smoothness: The smoothness metric for the CoT-based reward function is significantly lower, indicating more stable and consistent performance.

These findings suggest that the CoT-based reward function not only improves the agent's performance but also enhances its stability and efficiency. The higher mean fitness scores indicate better task completion, while the lower smoothness values suggest more consistent behavior over time.

In conclusion, the CoT-based reward function outperforms the benchmark methods in terms of both performance and stability, providing a robust framework for reward engineering in reinforcement learning tasks.

C. The effectiveness of CoT

In this section, we delve into the comparative analysis of the detailed CoT approach against the non-CoT method in the context of optimization search processes.

The experimental results highlight the improvements when incorporating CoT reasoning into the reward engineering process. As illustrated in Fig. 5, the CoT approach exhibits rapid convergence with fewer iterations required to reach high fitness values. Data points for the CoT method are predominantly located in the upper-left region of the graph, indicating efficient convergence to superior solutions within a shorter timeframe compared to the non-CoT method, which shows more scattered data points and requires more iterations to achieve similar or lower fitness levels.

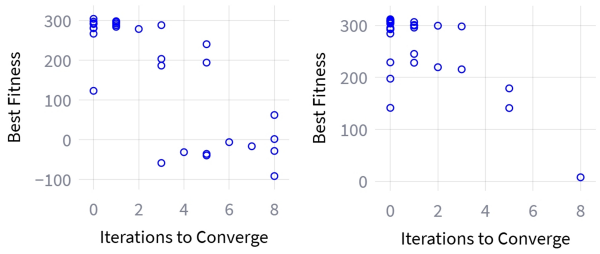

Fig. 5. Comparison of Convergence Speed vs Best Fitness between non-CoT (left) and CoT (right) approaches.

In addition, Fig. 6 shows that the data points using the CoT method are mainly clustered in the upper left part, indicating that the algorithm can also achieve higher final performance under the condition of low standard deviation (i.e. high stability). In contrast, non CoT methods have a wider range of data points, especially in the high standard deviation

range. This shows that with the guidance of CoT thinking chain, the fluctuation of the algorithm is reduced, and the final performance results are more consistent and reliable.

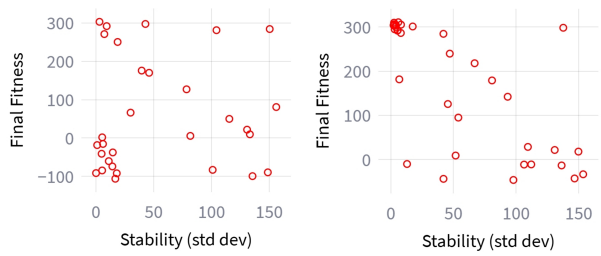

Fig. 6. Comparison of Stability vs Final Performance between non-CoT (left) and CoT (right) approaches.

The data presented in Table III further substantiates the aforementioned observations. The mean fitness values are consistently higher across all iterations with CoT, indicating more effective optimization. The common range, which shows the interval with the highest frequency of fitness values, is predominantly positive for CoT, contrasting with the negative ranges often seen in non-CoT methods. This highlights CoT's role in achieving better and more stable performance outcomes.

TABLE III
COMPARISON OF NON-COT AND COT RESULTS OVER ITERATIONS.

Iteration	Mean Fitness		Common Range	
	non-CoT	CoT	non-CoT	CoT
0	1.38	122.34	[-92.1, -14.2]	[217.5, 309.9]
1	58.61	188.20	[-91.7, -13.7]	[226.9, 309.8]
2	67.17	202.37	[-106.9, -24.6]	[226.4, 309.8]
3	87.05	201.39	[-92.1, -14.5]	[237.8, 312.0]
4	66.11	206.04	[-138.3, -50.0]	[241.3, 307.5]
5	83.38	169.15	[222.6, 303.2]	[242.1, 309.9]
6	97.31	179.22	[221.3, 303.2]	[230.1, 310.6]
7	74.34	159.73	[-91.5, -14.8]	[239.3, 310.6]

Overall, the experimental results indicate that the incorporation of CoT reasoning provides benefits in terms of acceleration of convergence, enhancement of performance, and improvement of result stability. These enhancements contribute to the development of more robust and efficient RL algorithms.

CONCLUSION

This study proposes the efficacy of utilizing Chain of Thought-enabled Large Language Models for generating reward functions in Reinforcement Learning environments. Through a detailed case study on the Bipedal-Walker environment, we propose how CoT reasoning facilitates the creation of nuanced and effective reward structures, leading to superior agent performance. The ablation studies further support the proposed contribution of CoT reasoning to reward function quality. This approach holds significant potential for streamlining reward engineering processes and enhancing RL applications across various domains.

REFERENCES

[1] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[2] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[3] 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.

[4] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv preprint arXiv:1909.07528*, 2019.

[5] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[6] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[7] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: A new approach of reward shaping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 931–15 941, 2020.

[10] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.

[11] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103500, 2021.

[12] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[13] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International conference on machine learning*. PMLR, 2016, pp. 49–58.

[14] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[15] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[16] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," *arXiv preprint arXiv:2303.00001*, 2023.

[17] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Automated dense reward function generation for reinforcement learning," *arXiv preprint arXiv:2309.11489*, 2023.

[18] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[19] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma, "Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics," *arXiv preprint arXiv:2309.06687*, 2023.

[20] 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.

[21] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. A. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1950–1965, 2022.

[22] 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.

[23] L. Pan, M. Saxon, W. Xu, D. Nathani, X. Wang, and W. Y. Wang, "Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies," *arXiv preprint arXiv:2308.03188*, 2023.
[24] X. Wang and D. Zhou, "Chain-of-thought reasoning without prompting," *arXiv preprint arXiv:2402.10200*, 2024.
[25] X. Wu, H. Zhao, Y. Zhu, Y. Shi, F. Yang, T. Liu, X. Zhai, W. Yao, J. Li, M. Du *et al.*, "Usable xai: 10 strategies towards exploiting explainability in the llm era," *arXiv preprint arXiv:2403.08946*, 2024.