PRACT: Optimizing Principled Reasoning and Acting of LLM Agent

Zhiwei Liu*, **Weiran Yao**†, **Jianguo Zhang, Rithesh Murthy, Liangwei Yang, Zuxin Liu, Tian Lan, Ming Zhu, Juntao Tan, Shirley Kokane, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese** and **Caiming Xiong**
Salesforce AI Research, USA

Abstract

We introduce the Principled Reasoning and Acting (PRAct) framework, a novel method for learning and enforcing action principles from trajectory data. Central to our approach is the use of text gradients from a reflection and optimization engine to derive these action principles. To adapt action principles to specific task requirements, we propose a new optimization framework, Reflective Principle Optimization (RPO). After execution, RPO employs a reflector to critique current action principles and an optimizer to update them accordingly. We develop the RPO framework under two scenarios: Reward-RPO, which uses environmental rewards for reflection, and Self-RPO, which conducts self-reflection without external rewards. Additionally, two RPO methods, RPO-Traj and RPO-Batch, is introduced to adapt to different settings. Experimental results across four environments demonstrate that the PRAct agent, leveraging the RPO framework, effectively learns and applies action principles to enhance performance.

1 Introduction

Large language model (LLM) agents enable the action execution (Gravitas, 2023; Goodman, 2023; Yao et al., 2023a; Wang et al., 2023a) and consecutive reasoning ability (Nakajima, 2023; Shinn et al., 2023; Yao et al., 2023b) of LLM. Specifically, an LLM agent has both memory (Shinn et al., 2023; Li et al., 2023; Liu et al., 2024) and action space (Chase, 2023; Wu et al., 2023; Liu et al., 2023). Adding those information into prompt extends the inference of LLM to be multi-turn action execution. Therefore, an LLM agent is able to decide next actions based on its previous execution observations (Wang et al., 2023b; Xu et al., 2023; Goodman, 2023; Song et al., 2023).

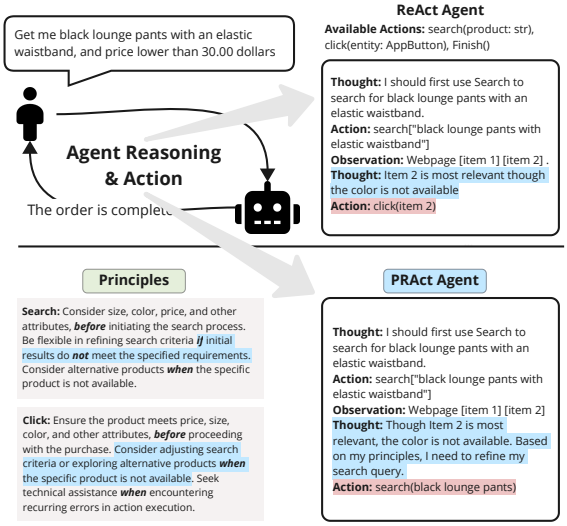

Figure 1: Comparison of ReAct and PRAct agents.

Optimizing the reasoning framework (Yao et al., 2023a; Liu et al., 2023; Wang et al., 2023b) of agent is crucial in generating correct action execution. As of now, customizing an LLM agent with existing open-source packages (Liu et al., 2024; Wu et al., 2023; Chase, 2023; Liu, 2022) requires the designing of action spaces, such as function calls (Patil et al., 2023) and code execution (Wu et al., 2023, 2024). Along with a well-designed agent reasoning framework, i.e. the prompts of agent, an LLM is able to consecutively generate correct actions. ReAct (Yao et al., 2023a) framework achieves wide successes via adding one-step *think* actions to enhance the reasoning ability of an agent. Additionally, Reflection (Shinn et al., 2023; Yao et al., 2023b; Paul et al., 2023) mechanism is proposed to improve the agent self-correction capability. *Plan* (Xu et al., 2023; Liu et al., 2023) before execution is also verified to be beneficial.

Despite many successes, agent execution can fail to make decisions when faced with contradictory observations, particularly during the execution of long-step tasks. To address it, we propose a new

*zhiweiliu@salesforce.com
†Equal contribution.

type of reasoning strategy, *PRAct*, for the LLM agent. Intuitively, we associate each action with principles that describe the conditions for using that action. During execution, an agent can check these principles before generating the next action. Compared to simple action descriptions, principles provide more detailed conditions on when to use the action and offer specific instructions on how to generate the parameters for an action. We demonstrate the benefits of PRAct in Figure 1 via comparing with ReAct agent in WebShop (Yao et al., 2022) where an agent uses search and click actions to interact with a shopping website. The ReAct agent searches a query and, despite item 2 not having the available color, still clicks it as it appears most relevant. In contrast, the PRAct agent refines the search based on both search and click principles. Consequently, the PRAct agent decides to search with an improved query, enhancing its decision-making process.

To reduce the labor involved in prompt design and to cover more complex scenarios, we propose a new principle optimization framework, Reflective Principle Optimization (RPO). RPO operates in three stages: execution, reflection, and optimization. During the execution stage, an agent performs tasks using predefined or null principles and memorizes the task trajectories. In the reflection stage, the agent reviews its task executions, evaluating how actions were selected and whether they met the task requirements. Finally, in the optimization stage, an optimizer refines principles to enhance agent performance. We investigate two optimization methods: RPO-Traj, which individually optimizes principles for each trajectory, and RPO-Batch, which concatenates all reflections in a batch for optimization.

We summarize our contributions as follows: 1) PRAct is the first work that considers the action principles for LLM agent; 2) we propose two optimization methods to adapt the principles to tasks.

2 PRAct: Optimizing Principled Reasoning and Acting

2.1 Formulation

Given a task query, an agent is able to consecutively execute actions $[a_1, a_2, \ldots, a_n]$ and collects observations $[o_1, o_2, \ldots, o_n]$ from environments, where o_i is the execution results of a_i. A policy function $\pi(a_t|c_t)$ predicts the next action a_t given the execution trajectory context $c_t = [(a_1, o_1), (a_2, o_2), \ldots, (a_{t-1}, o_{t-1})]$. An Executor agent utilizes a language model to determine the policy function. It requires textual trajectory information for the prompt Intrinsically, those context information are text-based, including action names, action parameters and observations.

PRAct constraints the reasoning of LLM to follow a set of principles $\mathcal{P}$ as follows:

$$\pi(a_t|c_t) = \text{Executor}(a_t|\mathcal{T}(c_t); \mathcal{P}), \quad (1)$$

where $\mathcal{T}$ is the prompt template to organize context information and the principles $\mathcal{P}$ are guidelines that help shape the decision-making process of an LLM agent. Principles provide instructions on the usage of the action such as how to generate parameters for the action. Additionally, principles reduce the set of potential actions by eliminating those that do not conform to the defined guidelines, thereby narrowing the search within the action space. In this paper, we simplify the principles space to be the same as actions space, *i.e.* each $a_i \in \mathcal{A}$ associated with a $p_i \in \mathcal{P}$.

2.2 Reflective Principle Optimization (RPO)

Although the principles could be predetermined, as in the action descriptions, it is challenging to comprehensively cover all possible conditions without an automatic optimization paradigm. Therefore, we propose a new algorithm, Reflective Principle Optimization (RPO), to adapt principles for complex scenarios. RPO operates in three stages: 1) Execution, 2) Reflection, and 3) Optimization.

2.2.1 Execution

Given a set of tasks, the executor agent performs actions based on the current set of principles, collecting observations from the environment. This stage involves prompting the LLM agent to generate actions, which regressively calls Eq. (2) until reaching the final actions or maximum steps. Given a task query q, we denote the trajectory as $c_q = [(a_q^{(1)}, o_q^{(1)}), (a_q^{(2)}, o_q^{(2)}), \cdot, (a_q^{(n)}, o_q^{(n)})]$. Note that those actions may be some inner actions, such as *think* or *plan* (Yao et al., 2023a; Liu et al., 2024) , which do not forward to the environment and are associated with a default or null observation. Executor collects a set of trajectory context sequences $\mathcal{C}$ for those queries $\mathcal{Q}$ during execution stage.

2.2.2 Self-Reflection

After executing the actions, a reflector agent reflects on trajectories $\mathcal{C}$ by analyzing the collected

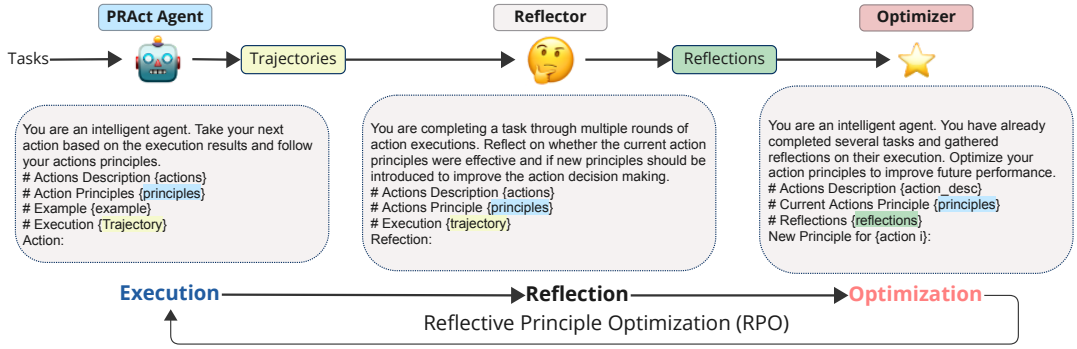

Figure 2: **PRAct and RPO overview.** Each iteration three stages: execution, reflection and optimization. During execution, an agent executes tasks with previous principles. The trajectories are saved. Then, the agent reflects on those tasks executions. Finally, the agent leverages those self-reflection results to optimize the principle.

Table 1: Overall comparison results. **Bold** denotes the best performance.

	GPT-3.5-turbo				GPT-4-turbo			
	WebShop	Academia	Movie	Weather	WebShop	Academia	Movie	Weather
Act	0.4542	0.5304	0.5483	0.5869	0.5257	0.6704	0.5875	0.6882
ReAct	0.4742	0.5504	0.5416	0.5973	0.5667	0.7428	0.5583	0.6990
Reflexion	0.5539	0.6024	0.5728	0.5876	0.5723	0.7796	0.6072	0.7197
ExpeL	0.5823	0.6318	0.6215	0.6475	0.6329	0.8084	0.6847	0.7583
PRAct-T	**0.6012**	0.6798	0.6595	0.6953	0.6323	**0.9207**	0.7132	0.7796
PRAct-B	0.5904	**0.7396**	**0.6625**	**0.7042**	**0.6413**	0.8254	**0.7250**	**0.8331**

observations. This stage involves evaluating the effectiveness of the actions in each trajectory and the adherence to the principles as follows:

$$r_q = \text{REFLECTOR}(c_q, \mathcal{P}), \tag{2}$$

for all $c_q \in \mathcal{C}$. The reflection process identifies conditions or guidelines where the principles need adjustment to better handle the observed tasks. If an environment provides rewards toward the execution, it is a reward-based reflector aligning the executions with reward feedback. Instead, if no rewards present for execution, it is a self-reflector.

2.2.3 Optimization

Based on the reflection results, we leverage the generation ability of LLM to refine the principles for improving the performance of agent in similar future scenarios. This stage involves refining the principles to better align with the observed conditions and enhance decision-making. We investigate two types of optimization methods.

RPO-Traj. This approach individually considers each trajectory and its reflection to optimize principles. Then a batch of principles are summarized as a new set of tailored principles $\mathcal{P}^*$. We formulate RPO-Traj as follows:

$$\mathcal{P}^* = \sum_{\mathcal{Q}} \text{OPT}(r_q, \mathcal{P}), \tag{3}$$

where $\sum_{\mathcal{Q}}$ denotes a summarizor of all principles generated from optimizer OPT for all queries $\mathcal{Q}$.

RPO-Batch. We use a prompt template to concatenate all the reflections in a batch. Then the optimizer directly generates new principles via considering all those reflections, which is formulated as follows:

$$\mathcal{P}^* = \text{OPT}(\text{CONCAT}\{r_q | q \in \mathcal{Q}\}, \mathcal{P}), \tag{4}$$

where CONCAT denotes using a prompt template to concat those reflections. In comparison, RPO-Traj requires generating principles for $|Q|+1$ times, while RPO-Batch only needs one time principles generation but with $|Q|$ times longer context length. Hence, long context reasoning ability is necessary for an optimizer in RPO-Batch method.

3 Experiment

3.1 Experiment Setup

Baselines. We compare our PRAct agent with existing Act, ReAct (Yao et al., 2023a), Reflexion (Shinn et al., 2023) agent reasoning methods

and Expel (Zhao et al., 2024) prompt optimization framework. In this paper, we employ GPT-3.5-Turbo-0125 and GPT-4-Turbo-2024-04-09 (OpenAI, 2023) as two foundation LLMs. And for simplicity, the executor, reflector and optimizer in PRAct are of the same language model.

Benchmarks and Evaluation. Following AgentBoard (Ma et al., 2024), we evaluate PRAct agent on three tool environments and one WebShop environment. Tool environments support the designing of WEATHER, MOVIE, and ACADEMIA agents. Tasks are 60 queries and actions are a set of function calls. The reward score is the recall of ground truth actions. Webshop environment is a web browser simulation. Agent performs either *search* and *click* actions to complete 251 online shopping tasks. Reward is attributes coverage ratio between final shopped items and ground truth item.

3.2 Optimization setup

For optimizing the WebShop agent with a Reward-based reflector, we randomly split the query tasks into training, validation, and test tasks with a ratio of 3:1:1. During each training step, we sample a batch of training tasks to execute and use RPO to optimize the principles. Performance on validation tasks is used for early stopping, and results are reported on test tasks. For tool agents, we use a self-reflector without rewards, making reflection tasks the same as test tasks. Since there is no ground truth, no data leakage problem exists. We tune the training batch size in [10,20,40] for WebShop and [2,4,6] for tool environments.

3.3 Experiment Results

Overall Performance. We present comprehensive comparisons of our methods against the agent baselines in Table 1. PRAct-T and PRAct-B are our methods with RPO-Traj and RPO-Batch optimization methods, respectively. We observe consistently better performance of PRAct agent, which demonstrates the effectiveness of principles in improving agent performance. Between the two optimization methods, *i.e.* PRAct-T and PRAct-B, PRAct-B generally performs better than PRAct-T. The reason is that summarizing principles from a batch of reflections enables potential reasoning across trajectories. However, PRAct-T outperforms PRAct-B due to the potential weaker long context understanding ability of GPT-3.5-Turbo, which indicates batch-wise optimization is more suitable for larger models.

Reflector	GPT-3.5	GPT-4
Self-T	0.5871	0.6172
Self-B	0.5763	0.6238
Reward-T	**0.6012**	0.6323
Reward-B	0.5904	**0.6413**

Table 2: Different reflectors of PRAct. *Self* and *Reward* stand for self and reward-based reflectors, respectively. T and B denote RPO-Traj and RPO-Batch, respectively.

An additional variant is *PRAct with self-reflector* on Webshop. We compare it on both RPO-T and RPO-B optimization methods, and report the results in Table 2. Compared with both results, reward-based reflector, demonstrates its superiority in optimizing principles with rewards.

Optimization Curve. We present the training curves in Fig. 3. Although at each step, we did not pick the best principle out of the sampled action principles on the validation set, we still observe consistent improvement over time. Notably, with action principles optimized by PRAct, LLM agents under GPT-3.5-Turbo can match the performance of GPT-4-turbo in Webshop environment.

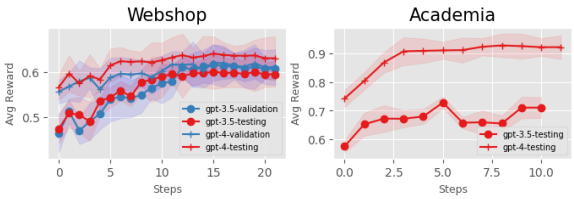

Figure 3: Training curves in Webshop and Academia with different LLMs and data splits. The reported scores are the average across 5 random seeds.

4 Conclusion

We propose a novel agent reasoning framework, PRAct, which provides principles of actions and thus benefits the action understanding of agent. Besides, we introduce two optimization algorithm, RPO-Traj and RPO-Batch for adapting the action principles with task executions. Experimental results on four environments demonstrates the effectiveness of PRAct framework. And the training curve illustrates the learning efficacy of RPO. In conclusion, PRAct opens a new discussion on how to regularize the agent actions while RPO shades the light on how to optimize the agent prompts.

References

Harrison Chase. 2023. Langchain. https://github.com/hwchase17/langchain.

Noah Goodman. 2023. Meta-prompt: A simple self-improving language agent. *noahgoodman.substack.com*.

Significant Gravitas. 2023. Autogpt. https://github.com/Significant-Gravitas/Auto-GPT.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jerry Liu. 2022. LlamaIndex.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, et al. 2023. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents. *arXiv preprint arXiv:2308.05960*.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Liangwei Yang, Zuxin Liu, Juntao Tan, Prafulla K. Choubey, Tian Lan, Jason Wu, Huan Wang, Shelby Heinecke, Caiming Xiong, and Silvio Savarese. 2024. Agentlite: A lightweight library for building and advancing task-oriented llm agent system. *Preprint*, arXiv:2402.15538.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *Preprint*, arXiv:2401.13178.

Yohei Nakajima. 2023. Babyagi. https://github.com/yoheinakajima/babyagi.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023a. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Yu Wang, Zhiwei Liu, Jianguo Zhang, Weiran Yao, Shelby Heinecke, and Philip S Yu. 2023b. Drdt: Dynamic reflection with divergent thinking for llm-based sequential recommendation. *arXiv preprint arXiv:2312.11336*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. 2024. Stateflow: Enhancing llm task-solving through state-driven workflows. *arXiv preprint arXiv:2403.11322*.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023a. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2023b. Retroformer: Retrospective large language agents with policy gradient optimization. *Preprint*, arXiv:2308.02151.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 19632–19642. AAAI Press.