MultiPrompter: Cooperative Prompt Optimization with Multi-Agent Reinforcement Learning

Dong-Ki Kim[1]
dkkim@lgresearch.ai

Sungryull Sohn[1]
srsohn@lgresearch.ai

Lajanugen Logeswaran[1]
llajan@lgresearch.ai

Dongsub Shim[1]
dongsub.shim@lgresearch.ai

Honglak Lee[1,2]
honglak@lgresearch.ai

Abstract

Recently, there has been an increasing interest in automated prompt optimization based on reinforcement learning (RL). This approach offers important advantages, such as generating interpretable prompts and being compatible with black-box foundation models. However, the substantial prompt space size poses challenges for RL-based methods, often leading to suboptimal policy convergence. This paper introduces MultiPrompter, a new framework that views prompt optimization as a cooperative game between prompters which take turns composing a prompt together. Our cooperative prompt optimization effectively reduces the problem size and helps prompters learn optimal prompts. We test our method on the text-to-image task and show its ability to generate higher-quality images than baselines.

1 Introduction

Foundation models are now an integral part of our daily lives, finding applications across various tasks and domains [1–3]. The driving force behind their widespread adoption is prompting. Unlike fine-tuning, which involves a resource-intensive process of updating numerous model parameters for a specific task, prompting effectively guides the model's behavior by refining initial prompts [4–6]. With the growing availability of black-box models, prompting has emerged as an essential tool for interacting with foundation models.

An important goal in prompting is automated prompt optimization, removing the need for laborious manual trial-and-error efforts. Reinforcement learning (RL) [7] presents a promising solution for achieving this goal by discovering prompts that outperform manually created ones through sequential optimization in the prompt space [8–13]. Notably, RL-based methods generate interpretable prompts and are compatible with black-box foundation models. These attributes provide distinct advantages over alternative approaches like soft prompts [14–17], which produce less interpretable prompts and require white-box access to the models. However, despite these exciting outcomes, we note that existing work suffers from the extensive size of the prompt space. This problem size is critical as it hinders effective exploration and generally leads to suboptimal policy convergence [18].

We propose a novel prompt optimization framework to address the challenge posed by the extensive prompt space. Our key idea is to view prompt optimization as a cooperative game between multiple prompters which take turns composing a prompt together (see Figure 1). In this new setting, prompters cooperatively decompose the prompt space into smaller subspaces and sequentially optimize their respective parts. As a result, our approach significantly reduces problem complexity compared to prior work that rely on a single prompter to optimize the entire prompt. To effectively learn

[1]LG AI Research [2]University of Michigan–Ann Arbor

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

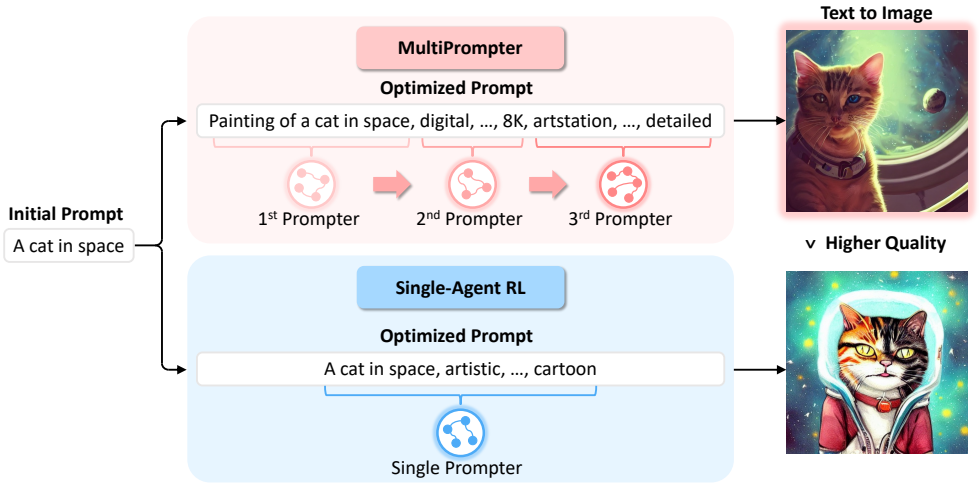

Figure 1: In MultiPrompter, a team of prompters learns to take turns optimizing a prompt together, generating higher-quality images compared to those produced by a single-agent RL method for a text-to-image task. The images are generated using Stable Diffusion [2].

cooperative policies within our framework, we develop a practical multi-agent RL algorithm, named *MultiPrompter*. Specifically, we enable each prompter to consider the behaviors of subsequent prompters through a centralized critic [19–22] designed for cooperative prompt optimization. We show that MultiPrompter generates more optimal prompts than those produced by baselines.

Our contribution. In summary, this paper makes the following main contributions:

- **Formulation of cooperative prompt optimization (Section 2).** We introduce a new cooperative game in which multiple prompters work as a team to enhance a prompt by sequentially optimizing it. This cooperative approach effectively reduces problem complexity in contrast to a single-agent RL approach and facilitates the process of finding optimal prompts.
- **Algorithm for learning cooperative prompt optimization (Section 3).** Learning multiple prompters in our setting requires each prompter to take into account the behaviors of others. Otherwise, undesirable outcomes can arise as prompters may greedily optimize a prompt. To address this, MultiPrompter develops an actor-critic framework with a centralized critic that considers the next prompter's actions to predict the value accurately.
- **Comprehensive evaluation of MultiPrompter (Section 4).** Our results highlight that MultiPrompter outperforms a single-agent RL baseline in the text-to-image generation task, obtaining higher rewards and the ability to optimize longer prompts. Additionally, we consider a variant of MultiPrompter, which applies a competitive game between prompters, and show that this competitive prompt optimization is less effective than our cooperative formulation.

2 Problem Statement: Cooperative Prompt Optimization Game

Overview. We introduce the new concept of cooperative prompt optimization: given an initial prompt x, a team of n prompters generates an optimized prompt that consists of multiple subprompts $y = (y^1; \ldots; y^n)$ (see Figure 1). Specifically, the first prompter optimizes the initial parts of a prompt and passes the baton to the next prompter as needed. Then, the second prompter continues the optimization from where the previous prompter left off. This process repeats until the last prompter finishes its turn or the length of an optimized prompt exceeds a token limit. The team's objective is to generate an optimized prompt such that it achieves a high score according to a performance metric.

Definition. We formally define a cooperative prompt optimization game between n prompters as a tuple $G_n = \langle \mathcal{I}; S; V; T; R \rangle$; $\mathcal{I} = (1; \ldots; n)$ is the set of n prompters; S is the prompt space; V is the vocabulary; $T : S \times V \to S$ is the deterministic prompt transition function; and R is the reward function. We also define an index $i \in \mathcal{I}$ that points to the active prompter which is currently taking its turn. At the beginning of the game, prompters are given an initial prompt $x = (x_1; \ldots; x_K)$ and the index i is reset to one. At each timestep t, the i-th prompter decides a discrete token $y_t \in V$ according to its stochastic policy $y_t \sim \pi^i(\cdot|x; y_{1:t-1}; \theta^i)$ parameterized by θ^i, where $y_{1:t-1} = (y_1; \ldots; y_{t-1})$.

An action y_t then yields the prompt transition from $(x; y_{1:t91})$ to $(x; y_{1:t})$. If y_t corresponds to an end-of-sequence token, then i-th prompter finishes generating its subprompt $\mathbf{y}^i$. The index i is also updated to pass the turn to the next prompter (i.e., $i \quad i+1$). The game ends when all prompters finish their turns or the size of an optimized prompt y is over the token limit T. The team shares a reward according to $R(x; y)$, which measures the quality of an optimized prompt y at the end of the game.

Benefit of problem size reduction. The natural outcome of our cooperative game is the generation of a decomposed optimized prompt, which consists of subprompts: $y = (\mathbf{y}^1; \ldots; \mathbf{y}^n)$. Thanks to this cooperative prompt decomposition, each prompter i in MultiPrompter simply has to search for the desired tokens within its subprompt $\mathbf{y}^i$, which contains fewer tokens than the full prompt y. As a result, we note that our formulation substantially reduces problem complexity compared to optimizing a prompt using single-agent RL methods [8, 23]:

$$\underbrace{\sum_{i=1}^{n} |V|^{|\mathbf{y}^i|}}_{\text{Multi prompter problem size}} \qquad \underbrace{|V|^{|y|}}_{\text{Single prompter problem size}} \tag{1}$$

where $|\cdot|$ denotes the size of a set. In the following section, we leverage this inherent benefit of our cooperative prompt optimization game and develop an algorithm for learning cooperative policies.

3 MultiPrompter: Learning Cooperative Prompt Optimization Policies

This section introduces our multi-agent learning algorithm, named MultiPrompter, designed to learn cooperative policies that decompose and optimize a prompt together. We first outline each prompter's objective in our cooperative prompt optimization game. We then detail our centralized critic that enables each prompter to consider the actions and learning of others, thus facilitating successful cooperation. We provide additional details, including pseudocode, in Appendix A.

MultiPrompter objective. The objective of each prompter's policy $\quad^i$ is to find policy parameters $\quad^i$ that maximize the expected return:

$$\max_i \mathbb{E}_{x;y \quad p(\cdot)} R(x; y) \;; \text{ where } p(x; y|\quad) = p(x) \prod_{i=1}^{n} \prod_{t=t^i_{\text{bos}}}^{t^i_{\text{eos}}} \quad^i(y_t|x; y_{1:t91}; \quad^i); \tag{2}$$

where t^i_{bos} and t^i_{eos} denote the beginning and end timesteps of $\mathbf{y}^i$, respectively. Leveraging REINFORCE [24], we derive a policy gradient with respect to the objective in Equation (2):

$$r \quad^i \mathbb{E}_{x;y \quad p(\cdot)} \sum_{t=t^i_{\text{bos}}}^{t^i_{\text{eos}}} \log \quad^i(y_t|x; y_{1:t91}; \quad^i) A^i_t(x; y) \;; \tag{3}$$

where $A^i_t(x; y)$ denotes the advantage function, which we will detail our choice in the next paragraph.

Centralized critic. The generalized advantage estimation [25] shows that the advantage function can be effectively estimated with low variance using a value function. A single-agent RL approach uses a value function $v(x; y_{1:t91}; \quad)$ parameterized by $\quad$ [9], but this form neglects the presence of other prompters, even though the reward is jointly affected by all of them. To consider policies and learning of subsequent prompters, MultiPrompter uses a value function $v^i(x; y_{1:t91}; \mathbf{y}^{i+1}; \quad^i)$ that enables a prompter i to consider a prompter $i+1$'s policy (see Figure 2). We have a design choice regarding the number of next prompters to consider in the value function, and we empirically find that including the next prompter's information results in effective training. Note that the centralized value function is only utilized during training, which takes additional information to accurately predict the value. Each prompter's policy remains decentralized, so MultiPrompter follows the centralized training with decentralized execution structure [19–22]. Lastly, we provide value function optimization and other related details in Appendix A.

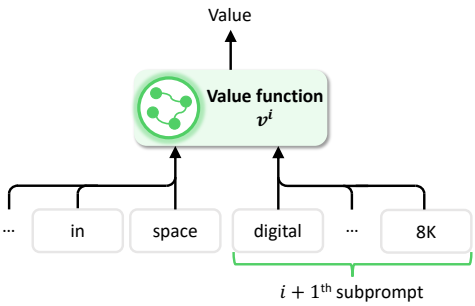

Figure 2: Our value function additionally considers a subprompt of the next prompter $\mathbf{y}^{i+1}$.

Algorithm	Multi-agent?	Collaborative?	Test reward
Manual prompt	✗	✗	90.68±0.06
Promptist	✗	✗	0.28±0.11
Competition	✓	✗	0.36±0.12
MultiPrompter	✓	✓	0.76±0.10

Table 1: Test performance across various methods. MultiPrompter achieves a statistically significant performance improvement through cooperative prompt optimization.

Figure 3: MultiPrompter's performance w.r.t. n shows a trade-off between problem size reduction and cooperation complexity.

4 Evaluation

Experiment setup. Following the experimental settings by Hao et al. [9], we use a reward function $R(x, y)$ that consists of the relevance score (i.e., measures the degree of relevance between an initial prompt x and an image generated by y) and the aesthetic score (i.e., measures aesthetical preference of an image generated by y over another image generated by x). We implement both policies and value functions using GPT-2 [26] and initialize them from the weights fine-tuned with manually engineered prompts [9]. We compare methods using the COCO dataset [27]. We refer to Appendix B for more details.

Baseline. We compare MultiPrompter with the following baselines:

- Manual prompt [9]. A fine-tuned method with human-engineered prompts from Lexica [28].
- Promptist [9]. A single-agent RL method that trains a GPT-2 prompter based on PPO [29].
- Competition. Our variant of MultiPrompter that applies competition between prompters [30].

We omit soft prompt methods [14–17] in our evaluation, because our work focuses on generating interpretable prompts without access to foundation models.

Question 1. How effective is MultiPrompter compared to a single-agent RL baseline?

Table 1 provides a summary of the test time performance. For the training performance across multiple seeds, we refer to Figure 5 in the Appendix. Our main observation is that MultiPrompter outperforms both the single-agent RL and manual prompt baselines. To understand how MultiPrompter generates more optimal prompts compared to Promptist, we examine the number of optimized tokens by each method. While Promptist converges to optimizing an average of 57.01 tokens, MultiPrompter optimizes an average of 69.46 tokens (refer to Figure 6 in the Appendix for an example). This result indicates that the single-agent RL approach generally suffers from the extensive prompt space, converging to a suboptimal policy that adds only a limited number of modifiers to the original prompt. In contrast, MultiPrompter successfully overcomes this challenge through cooperative prompt optimization and finds a greater number of effective modifiers compared to Promptist.

Question 2. What about posing prompt optimization as a competitive game?

When considering prompt optimization from a multi-agent perspective, cooperative optimization is not the only approach, but there is also the alternative of competitive prompt optimization. We consider a competitive setting in which each prompter optimizes a full prompt individually and then compares its optimized prompt to the output of another competing prompter (refer to Appendix B.1 for details). As Table 1 shows, we find that competitive prompt optimization produces a slight improvement over the single-agent RL method but it is not as effective as cooperative prompt optimization, primarily due to its lack of the ability to decompose the prompt space.

Question 3. How does MultiPrompter's performance change with respect to the number of prompters?

Figure 3 shows an analysis of MultiPrompter in relation to the number of prompters n. There are two notable observations. First, we observe a trend in which test performance increases, but then decreases after $n = 3$. This result suggests a general trade-off in cooperative prompt optimization: while the prompt space size reduces as n increases (as discussed in Section 2), the complexity of learning cooperation between prompters increases. Our future work includes incorporating recent advances in multi-agent RL [31–34] to effectively address the learning complexity with increasing n. Second, we note that MultiPrompter with any $n > 1$ performs better than a single-agent RL approach (i.e., $n = 1$).

Question 4. How important is it to learn the prompt decomposition and have the centralized critic?

MultiPrompter achieves cooperative prompt optimization by learning the dynamic decomposition of the prompt space. This process involves each prompter learning the appropriate moment to finish its turn and pass the baton to the next prompter, while taking into account the actions of the following prompter through the centralized critic. Table 2 presents an ablation analysis of MultiPrompter, examining its performance in learning a flexible prompt space decomposition versus manual decomposition (i.e., fixing each prompter's token limit as 1). We also study the impact of using the centralized critic in learning policies. Two notable observations emerge. First, without the centralized critic, each prompter cannot consider the behaviors of other prompters, resulting in ineffective collaboration. Second, while utilizing the centralized critic improves performance, the combination of this approach with the dynamic prompt space decomposition as in MultiPrompter leads to the best performance.

Learned decomposition?	Centralized critic?	Test reward
✗	✗	0.40±0.12
✓	✗	0.39±0.12
✗	✓	0.59±0.16
✓	✓	0.76±0.10

Table 2: Ablation study of MultiPrompter. By learning to collaboratively decompose the prompt space and employing the centralized critic to take into account the behaviors of the next prompter, MultiPrompter achieves the best performance.

5 Conclusion

In this paper, we have introduced MultiPrompter to address the extensive prompt space size in RL-based prompt optimization. The key idea is to learn multiple cooperative prompters that optimize a prompt together. We tested our method on various settings and showed that MultiPrompter consistently outperforms baseline approaches in the text-to-image domain.

References

[1] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[3] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[6] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9), jan 2023.

[7] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, second edition, 2018.

[8] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3369–3391, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[9] Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. arXiv preprint arXiv:2212.09611, 2022.

[10] Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. TEMPERA: Test-time prompt editing via reinforcement learning. The Eleventh International Conference on Learning Representations, 2023.

[11] Yihong Dong, Kangcheng Luo, Xue Jiang, Zhi Jin, and Ge Li. Pace: Improving prompt with actor-critic editing for large language models. arXiv preprint arXiv:2308.10088, 2023.

[12] Chengzhengxu Li, Xiaoming Liu, Yichen Wang, Duyi Li, Yu Lan, and Chao Shen. Dialogue for prompting: a policy-gradient-based discrete prompt optimization for few-shot learning. arXiv preprint arXiv:2308.07272, 2023.

[13] Hoyoun Jung and Kyung-Joong Kim. Discrete prompt compression with reinforcement learning. arXiv preprint arXiv:2308.08758, 2023.

[14] Guanghui Qin and Jason Eisner. Learning how to ask: Querying LMs with mixtures of soft prompts. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5203–5212, Online, June 2021. Association for Computational Linguistics.

[15] Xiang Lisa Li and Percy Liang. Pre x-tuning: Optimizing continuous prompts for generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597, Online, August 2021. Association for Computational Linguistics.

[16] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-ef cient prompt tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[17] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to ne-tuning across scales and tasks. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[18] Cosmin Paduraru, Daniel J. Mankowitz, Gabriel Dulac-Arnold, Jerry Li, Nir Levine, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: de nitions, benchmarks & analysis. Machine Learning Journal, 2021.

[19] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[20] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. Proceedings of the AAAI Conference on Arti cial Intelligence, 32(1), Apr. 2018.

[21] Shayegan Omidsha ei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P. How. Learning to teach in cooperative multiagent reinforcement learning. Proceedings of the AAAI Conference on Arti cial Intelligence, 33(01):6128–6136, Jul. 2019.

[22] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022.

[23] Marwa Abdulhai, Dong-Ki Kim, Matthew Riemer, Miao Liu, Gerald Tesauro, and Jonathan P. How. Context-specific representation abstraction for deep option learning. Proceedings of the AAAI Conference on Artificial Intelligence, 36(6):5959–5967, Jun. 2022.

[24] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3):229–256, May 1992.

[25] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438, 2018.

[26] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, pages 740–755, Cham, 2014. Springer International Publishing.

[28] The state of the art ai image generation engine. https://lexica.art/. Accessed: 2023-09-28.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[30] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In International Conference on Learning Representations, 2018.

[31] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 5571–5580. PMLR, 10–15 Jul 2018.

[32] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. Machine theory of mind. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 4218–4227. PMLR, 10–15 Jul 2018.

[33] Dong Ki Kim, Miao Liu, Matthew D Riemer, Chuangchuang Sun, Marwa Abdulhai, Golnaz Habibi, Sebastian Lopez-Cot, Gerald Tesauro, and Jonathan How. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 5541–5550. PMLR, 18–24 Jul 2021.

[34] Dong-Ki Kim, Matthew Riemer, Miao Liu, Jakob Foerster, Michael Everett, Chuangchuang Sun, Gerald Tesauro, and Jonathan P How. Influencing long-term behavior in multiagent reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 18808–18821. Curran Associates, Inc., 2022.

[35] Yu-han Chang, Tracey Ho, and Leslie Kaelbling. All learning is local: Multi-agent learning in global reward games. In S. Thrun, L. Saul, and B. Schölkopf, editors, Advances in Neural Information Processing Systems, volume 16. MIT Press, 2003.

[36] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18, page 2085–2087, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[37] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In International Conference on Learning Representations, 2020.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 8748–8763. PMLR, 18–24 Jul 2021.

[39] Aion-aesthetics. https://laion.ai/blog/laion-aesthetics/. Accessed: 2023-09-28.

A Additional MultiPrompter Details

A.1 Optimization

The advantage function $A^i_t(x; y)$ in Equation (3) can be effectively estimated using the generalized advantage estimation [25] using a value function:

$$A^i_t(x; y) = \sum_{l=0}^{} \quad {}^{jy^i j \quad 1} \quad {}^{l} \quad {}^{i}_{t+l} \tag{4}$$

$$ {}^{i}_{t+l} = \begin{array}{ll} R(x; y) \quad v^i(x; y_{1:t+l} \quad {}_1; y^{i+1}; \quad {}^i) & \text{if } t + l = t^i_{\text{eos}}; \\ v^i(x; y_{1:t+l}; y^{i+1}; \quad {}^i) \quad v^i(x; y_{1:t+l} \quad {}_1; y^{i+1}; \quad {}^i) & \text{else,} \end{array} \tag{5}$$

where we assume the discount factor $\quad = 1$ and sparse reward function $R(x; y)$. We update value function parameters by minimizing the standard squared-error loss with the target value $v_{\text{target};t}$:

$$L_v(\quad {}^i) = \mathsf{E}_{x;y \quad p(\ j \ )} \sum_{t=t^i_{\text{bos}}}^{t^i_{\text{eos}}} (v(x; y_{1:t \quad 1}; y^{i+1}; \quad {}^i) \quad v_{\text{target};t})^2 \ : \tag{6}$$

In this work, we apply PPO [29] to update policies and value functions with respect to Equation (3) and Equation (6), respectively.

A.2 Reward Engineering

Since MultiPrompter is a multi-agent learning approach, our work is also affected by the credit assignment issue in multi-agent RL [20, 35, 36], where certain agents do not actively participate in cooperation. In particular, we observe that a prompter may optimize most or the entire prompt by itself, thereby not providing opportunities to subsequent teammates. To effectively address this issue, we add the following cooperation reward, which computes the entropy with respect to the lengths of the subprompts, to the original reward:

$$R_{\text{cooperation}}(y) = H(y) = \sum_{i \, 2 \, I} (j y^i j = j y j) \log(j y^i j = j y j) \ = \log n: \tag{7}$$

Intuitively, this reward function encourages prompters to evenly decompose the prompt space such that they collectively optimize a prompt.

A.3 Pseudocode

Algorithm 1 MultiPrompter

Require: policy parameters $\boldsymbol{\theta} = (\theta^1, ..., \theta^n)$, value parameters $\boldsymbol{\phi} = (\phi^1, ..., \phi^n)$, token limit T

while not converged **do**
  # perform episode reset
  get an initial prompt $\boldsymbol{x} \sim p(\boldsymbol{x})$
  reset index $i = 1$ and timestep $t = 1$
  # start prompt optimization
  while $i \leq n$ **do**
    # select current token y_t
    if $t \leq T$ **then**
      sample current token from an active prompter $y_t \sim \pi^i(\cdot|\boldsymbol{x}, \boldsymbol{y_{1:t-1}}; \theta^i)$
    else
      set current token as an EOS token $y_t = y_{\text{eos}}$
    end if
    # update timestep t *and index* i
    update timestep $t \leftarrow t + 1$
    if y_t corresponds to an EOS token y_{eos} **then**
      update index $i \leftarrow i + 1$
    end if
  end while
  compute a team reward $\mathcal{R}(\boldsymbol{x}, \boldsymbol{y})$
  # train prompters
  for $i = 1, ..., n$ **do**
    train policy parameters θ^i according to Equation (3)
    train value function parameters ϕ^i according to Equation (6)
  end for
end while

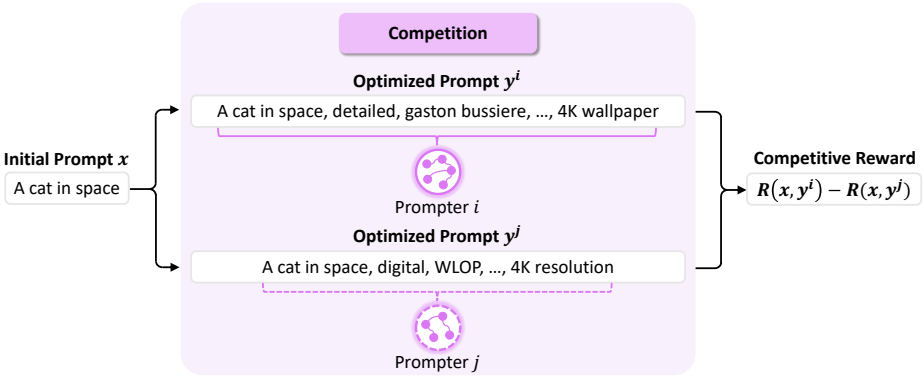

Figure 4: We present a competitive prompt optimization approach, where a prompter i competes against another prompter j by comparing their respective optimized prompts.

B Additional Evaluation Details

B.1 Competition Baseline Details

In this work, we contrast our cooperative prompt optimization approach with an alternative setting of competitive prompt optimization. Specifically, we design a competitive setting, where a prompter i generates its own full prompt y^i and then compares its response against another prompter j's optimized prompt y^j. A prompter i receives a competitive reward, defined as $R(x; y^i)$ $R(x; y^j)$, as a result of the comparison (see Figure 4). We apply the self-play technique [30, 37] to train a prompter that competes against its former copies and comparable skill levels. We also use the centralized critic with a form $v^i(x; y_{1:t91}; y^j;$ $^i)$ to enable a prompter i to consider another competition prompter j's policy.

B.2 Reward and Hyperparameter Details

Reward details. We follow [9] and use a reward function that measures the quality of an optimized prompt y. Specifically, the reward function $R(x; y)$ first computes the relevance score:

$$R_{\text{relevance}}(x; y) = \mathsf{E}_{img_y \quad M(y)} \min(20 f_{\text{CLIP}}(x; img_y) \quad 5{:}6; 0) \; ; \tag{8}$$

where img_y refers to an image generated according to y, M refers to a text-to-image model (e.g., Stable Diffusion [2]), and f_{CLIP} refers to the CLIP similarity function [38]. The relevance score measures the degree of relevance between img_y and the initial prompt x. The reward function also computes the aesthetic score that measures the aesthetical preference of img_y over img_x:

$$R_{\text{aesthetic}}(x; y) = \mathsf{E}_{img_x \quad M(x); img_y \quad M(y)} f_{\text{aesthetic}}(img_y) \quad f_{\text{aesthetic}}(img_x) \; ; \tag{9}$$

where $f_{\text{aesthetic}}$ refers to the aesthetic predictor [39]. Finally, the reward function sums the two scores: $R(x; y) = R_{\text{relevance}}(x; y) + R_{\text{aesthetic}}(x; y)$. For the case of cooperative prompt optimization, we also add the cooperation reward in Appendix A.2 during training: $R(x; y) = R_{\text{relevance}}(x; y) + R_{\text{aesthetic}}(x; y) +$ $R_{\text{cooperation}}(y)$, where denotes a weight.

Hyperparameter details. We report important hyperparameter values in our experiments:

Hyperparameter	Value
Number of prompters n	1,2,3,4,5
Batch size	256
Minibatch size	128
Stable Diffusion inference step	20
Token limit T	80
Learning rate	0.00001
Entropy weight	0.001
Discount factor	1.0
GAE	0.95
Cooperation reward weight	0.25

Table 3: Hyperparameter values used in our experiments.

B.3 Additional Result

Train performance.

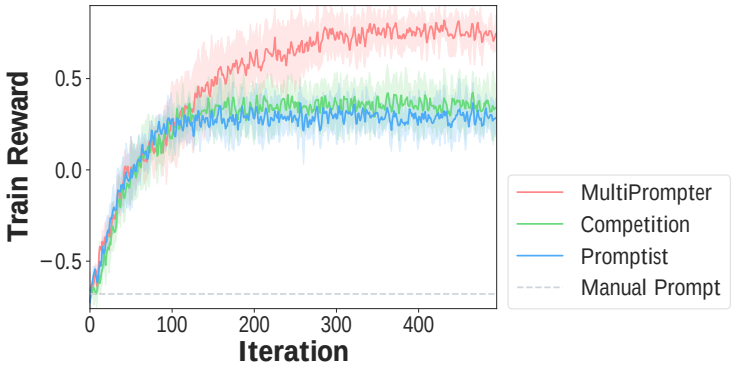

Figure 5: Training performance for each method. Thanks to our cooperative prompt optimization, MultiPrompter converges to a more optimal policy. The mean and standard deviation computed for 3 seeds are shown in the figure.

Test example.

40"$"#8%*3012 3)7 A3"#$%#3&#' '(3"#!)3$*3*$&&$+%3,+3!3-.!$)	
!"#$%&'($&#)*+,	-.%)"/012 3)&0
43)"2"5&6*/012 3)7 ! "#$%# &#''("#!) $* *$&&$+% ,+ ! -.!$)/ $+&)$-!&#0 #1#%!+&0 .$%.1('#&!$1#'0 '$%$&!1 2!$+&$+%0 !)&*&!&$,+0 -,+-#2& !)&0 *.!)2 3,-4*0 $114*&)!&$,+0 "(54*&$+ %#)!)' !+' !)&%#)6 0 7 8	43)"2"5&6*/012 3)7 0 9,*#2. :.)$*&$!+ ;#(#+'#-8#)0 <;=>0 ?)#!&.&!8$+%0 78)#*,14&$,+0 #@&)#6#1('#&!$1#'0 "#!4&$3410 #*&!"1$*.$+% *.,&0 !)&$*&$-0 .(2#))#!1$*&$-0 "#!4&$341 3!-# D#$%>),62&#) E&$>),62&#)

Figure 6: An example of an optimized prompt generated by a single-agent RL baseline and MultiPrompter. This example highlights that MultiPrompter adds a greater number of effective modifiers compared to the baseline. These images are generated using Stable Diffusion [2].