LLM-SMAC: Solving Multi-Agent Decision-Making Tasks via LLM Decision Tree Code Generation

Extended Abstract

Yue Deng
Zhongguancun Academy
Beijing, China
dengyue@zgci.ac.cn

Weiyu Ma
Chinese Academy of Sciences
Beijing, China
maweiyu2022@ia.ac.cn

Yuxin Fan
Xi'an Jiaotong University
Xi'an, China
fanyuxin@xjtu.stu.edu.cn

Ruyi Song
Zhejiang University
Hangzhou, China
3220105868@zju.edu.cn

Yin Zhang
Zhejiang University
Hangzhou, China
zhangyin98@zju.edu.cn

Haifeng Zhang
Chinese Academy of Sciences
Beijing, China
haifeng.zhang@ia.ac.cn

Jian Zhao
Zhongguancun Academy
Beijing, China
jianzhao@zgci.ac.cn

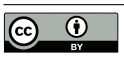
ABSTRACT

StarCraft Multi-Agent Challenge (SMAC) has become a widely used benchmark in multi-agent systems, where agents must control allied units to defeat enemy forces. Traditional MARL methods typically require millions of environment interactions to train parametric policies, which are often non-interpretable and exhibit limited transferability. In this paper, we introduce LLM-SMAC, a closed-loop Planner–Coder–Critic framework. Given task descriptions, the LLM planner first generates a decision-tree strategy, which is translated into executable code by the coder. The generated scripts are executed in the environment, and reward signals and runtime feedback are fed back to a critic module for self-reflection and iterative refinement. Through this closed-loop process, this mechanism progressively improves both strategy design and code implementation without large-scale environment exploration. We evaluate our method on the original SMAC tasks and the results show that LLM-SMAC can produce high-quality, interpretable decision trees with minimal interaction, while demonstrating strong transferability across homogeneous SMAC environments without modification.

KEYWORDS

Decision-making; Decision Tree Script; LLM; SMAC

ACM Reference Format:
Yue Deng, Weiyu Ma, Yuxin Fan, Ruyi Song, Yin Zhang, Haifeng Zhang, and Jian Zhao. 2026. LLM-SMAC: Solving Multi-Agent Decision-Making Tasks via LLM Decision Tree Code Generation: Extended Abstract. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/V1X2Y3Z4



Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/V1X2Y3Z4

1 INTRODUCTION

The StarCraft Multi-Agent Challenge (SMAC) [8], built on StarCraft II, is a standard benchmark for multi-agent reinforcement learning (MARL), featuring micromanagement tasks that require coordinated control of allied units. Algorithms such as VDN [9], QMIX [7], QPLEX [11], and MAPPO [12] have achieved strong empirical results on this benchmark.

However, existing MARL methods have key limitations. They often require millions of interactions, incur high computational cost, and produce black-box policies with limited interpretability. In addition, their transferability is weak, as models trained on one map typically need retraining to generalize to similar tasks.

In contrast, rule-based decision trees provide interpretable, white-box reasoning but rely heavily on expert knowledge and lack scalability. Recent advances in Large Language Models (LLMs) [1, 2, 5, 6, 10], particularly code-oriented LLMs [3, 4], enable reliable code generation and offer a potential bridge between rule-based and learning-based approaches. We posit that LLM agents can generate decision tree code from task descriptions and iteratively refine it using environmental feedback, thereby combining interpretability with scalability.

We propose LLM-SMAC, a Planner–Coder–Critic framework that generates and iteratively refines decision-tree code through closed-loop environmental feedback. Experiments on original SMAC maps show that LLM-SMAC achieves high win rates with minimal exploration, while producing interpretable strategies.

2 APPROACH

LLMs have acquired substantial knowledge of StarCraft II and related scripting frameworks during pretraining. Building on this foundation, our framework targets code generation for the `python-sc2` package, which provides a comprehensive wrapper for StarCraft II gameplay. Unlike traditional MARL approaches on `pysc2`, `python-sc2` relies on open-loop control scripts, requiring the LLM to anticipate

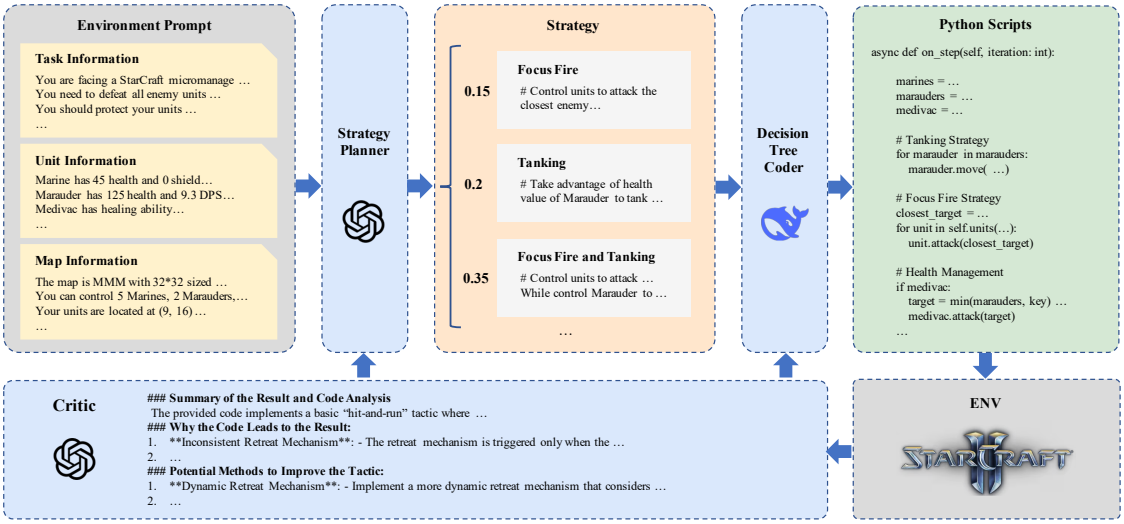


Figure 1: The Planner-Coder-Critic architecture. The Planner generates strategic skeletons from task information, the Coder produces executable Python scripts, and the Critic analyzes execution results to guide refinement.

and plan game dynamics prior to execution. Since planning, coding, debugging, and refinement impose heavy demands on a single model, we decompose the decision-tree generation process into three modules: a strategy planner, a code generator, and a critic.

As shown in Figure 1, for each SMAC task, unit and map information are formatted as an environment prompt for the planner LLM, along with previously learned skills and historical strategies. The planner outputs a strategy skeleton, including skill definitions and usage conditions in a decision-tree form. The coder then translates this structure into executable Python scripts and iteratively tests them on SMAC maps. Execution feedback, such as stack traces, win rates, scores, and damage statistics, is passed to the critic, which analyzes performance or bugs and provides suggestions for strategy refinement or code correction. The critic also determines whether the next iteration should revise the high-level plan or improve the implementation, forming a closed-loop workflow.

3 RESULTS

We evaluate LLM-SMAC on the StarCraft Multi-Agent Challenge (SMAC) benchmark spanning three difficulty categories. We use the DeepSeek-Coder-V2.5-236B) [3] as the backbone for all three modules. The results are shown in Table 1. Each script is evaluated over 10 episodes with different random seeds. We report win rate as the primary metric, the planning rounds and the coding rounds after planning.

Map	Win Rate	Plan Rounds	Code Rounds
3m	100%	2	1.4
8m	100%	5	1.0
25m	100%	32	1.2
8m_vs_9m	90%	22	2.0
10m_vs_11m	100%	19	1.0
27m_vs_30m	90%	35	1.2
2s3z	100%	31	2.0
3s5z	100%	22	3.0
1c3s5z	100%	11	2.6
2m_vs_1z	100%	4	1.0
3s_vs_3z	100%	3	1.0
3s_vs_4z	100%	14	1.2
3s_vs_5z	100%	14	1.4
2c_vs_64zg	100%	18	1.2
corridor	100%	17	1.0
2s_vs_1sc	100%	23	1.2
MMM	100%	15	3.6
bane_vs_bane	100%	14	2.4
so_many_baneling	100%	11	1.0
5m_vs_6m	0%	-	-
3s5z_vs_3s6z	0%	-	-
MMM2	0%	-	-
6h_vs_8z	0%	-	-

Table 1: LLM-SMAC achieves near-perfect win rate on 19 maps with interpretable strategies. The "-" indicates the pipeline failed to find winning scripts within iteration limits.

4 CONCLUSION

We propose LLM-SMAC, a closed-loop Planner–Coder–Critic framework for solving SMAC tasks through LLM-driven decision-tree generation. A large LLM iteratively plans strategies, generates executable code, and refines both design and implementation using environmental feedback. Experiments demonstrate that LLM-SMAC produces high-quality, interpretable decision trees with minimal exploration and strong transferability across tasks. In the future, the LLM-SMAC can be leveraged to generate adversarial decision scripts and construct an MARL training environment. In the future, the LLM-SMAC can be leveraged to generate adversarial decision scripts and provide a new MARL training environment.

REFERENCES

[1] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).

[2] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. arXiv:2103.10360 [cs.CL]

[3] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming–The Rise of Code Intelligence. *arXiv preprint arXiv:2401.14196* (2024).

[4] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186* (2024).

[5] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023). https://api.semanticscholar.org/CorpusID:257532815

[6] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[7] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv:1803.11485 [cs.LG] https://arxiv.org/abs/1803.11485

[8] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).

[9] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[10] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[11] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. arXiv:2008.01062 [cs.LG] https://arxiv.org/abs/2008.01062

[12] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. arXiv:2103.01955 [cs.LG] https://arxiv.org/abs/2103.01955