TwoStep: Multi-agent Task Planning using Classical Planners and Large Language Models

David Bai*[1], Ishika Singh*[1], David Traum[1], Jesse Thomason[1]

***Abstract*— Classical PDDL planning guarantees valid action sequences but struggles with concurrency unless domains are heavily modified by human experts. Large language models (LLMs) for planning excel at commonsense subgoal decomposition, yet lack execution guarantees. We combine the strengths of these systems by using LLMs to approximate multi-agent decompositions for single-agent PDDL planners, reducing both planning time and execution steps compared to direct multi-agent PDDL and single-agent baselines. Empirically, LLM-inferred subgoals match human expert decompositions while preserving plan correctness. Website and resources at glamor-usc.github.io/twostep**

I. Introduction

Multi-agent PDDL planning requires domain modification by a human expert and faces exponential search complexity. We investigate whether LLMs can exploit the semantic information in single agent expert-written planning domains to propose agent-specific subgoals for achieving global objectives. Our experiments span both symbolic and embodied domains.

Classical planning. Classical planning algorithms operate on finite, deterministic, fully observable states, guaranteeing a plan if one exists. Planning Domain Description Language (PDDL) and Answer Set Programming (ASP) are popular specification formats for such tasks [1], [2], [3], [4].

Planning with LLMs. Several works employ LLMs for agent planning [5], [6], but correctness is not guaranteed due to LLM stochasticity. Others combine LLMs with programmatic plan generation [7], [8], [9] or domain-specific heuristics [10], [11], though success is not assured. Recent efforts explore multi-agent code generation [12] or dialogue-based coordination [13]. In contrast, we use LLMs to infer commonsense goal decompositions, letting a PDDL solver ensure plan correctness.

II. Multi-agent Planning Method: TwoStep

We convert an N-agent problem into N single-agent subproblems via LLM-based subgoal decomposition. Specifically, we consider $N-1$ `helper` agents and one `main` agent. For a problem P with initial state i and goal g, each `helper` agent h generates a subgoal g_h and a plan $\pi_h = \prod(i, g_h)$, updating the state i_{h+1} after execution. Finally, the `main` agent plans from i_{N-1} to achieve the original goal g. We implement subgoal generation in two steps: English subgoal creation (subgoal generator) and translation to PDDL form (subgoal translator). Our approach, TwoStep (Figure 2), hypothesizes that LLMs can infer partially independent subgoals, enabling parallel action while preserving plan feasibility.

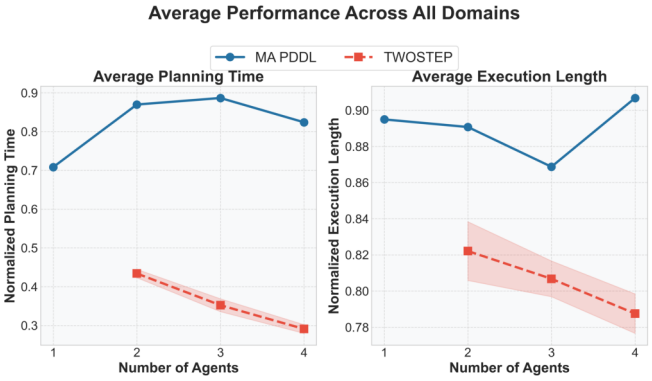

Fig. 1: Results comparing vanilla planning from SA PDDL ($N=1$) and MA PDDL ($N \geq 1$) to TwoStep as an average of normalized results across 5 symbolic domains (lower is better for both metrics). The shaded background is the standard deviation across 3 runs for TwoStep. All averages are across 20 tasks, except MA PDDL in the `TERMES` domain, which failed to produce plans for 5/20 tasks with 4 agents.

a) Subgoal Generation (subgoal generator and subgoal translator): We provide an in-context example problem and plan, plus two sample subgoals demonstrating independent progress and resource release. For a new problem P^d, the LLM iteratively proposes subgoals for $N-1$ `helper` agents or outputs 'None' if no additional helpers are needed. Subgoals are then translated by prompting the LLM with the domain description, initial state, and an example mapping.

b) Editing the Initial State for the `main` Agent.: We update the environment state after each `helper` agent's actions, ensuring the next agent (or `main` agent) sees a consistent state. Agent-specific conditions remain unaltered, allowing each agent to begin from its own valid initial configuration.

c) Multi-agent Plan Execution.: All agent plans are then executed in a shared environment. Symbolic domains use a dynamic programming exceution approach that is optimal for 2 agents and approximates optimal execution for ≥ 3, while embodied experiments are executed in AI2THOR simulator [14].

III. Results

We evaluate TwoStep against single-agent (SA PDDL) and multi-agent (MA PDDL) PDDL planning across 5 symbolic domains, and four long-horizon tasks in an embodied domain. We present our overall symbolic domain results in

*Equal Contribution
[1]Computer Science Department, University of Southern California {dmbai, ishikasi, traum, jessetho}@usc.edu

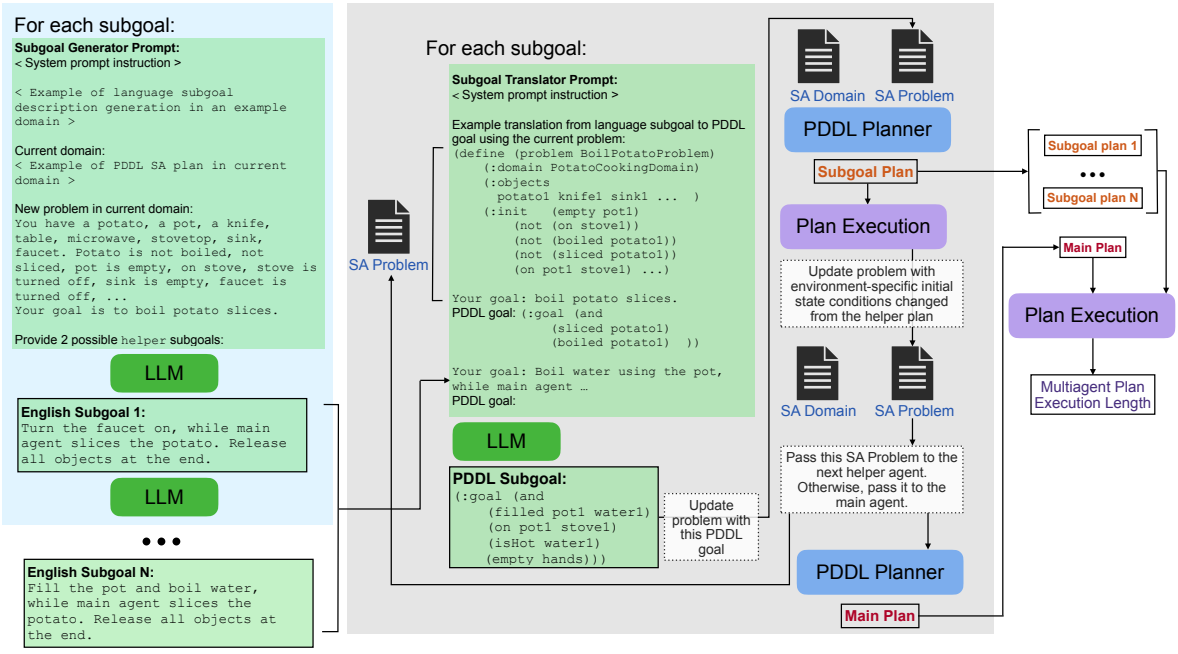

Fig. 2: TwoStep pipeline. $N-1$ `helper` agents each complete a partially independent subgoal to reduce steps for the `main` agent. All resulting plans run with partial parallelization to shorten overall execution length.

this abstract. For detalied results on symbolic and embodied domains, please refer to full paper [15].

Approaches. SA PDDL: Single-agent vanilla PDDL planning with agent-specific actions. MA PDDL: Multi-agent PDDL planning (2–4 agents), with expert modified single agent domains. TwoStep: A multi-agent pipeline with LLM-based subgoal decomposition among `helper` and `main` agents (Figure 2), executed as single-agent PDDL subplans in parallel. TwoStep requires no domain file modifications.

Evaluation Metrics. We measure *planning time* (seconds) and *execution length* (environment steps). All methods succeed in reaching the goal, so we omit success rates. For TwoStep, planning time includes (1) the solver time for each `helper` and `main` agent and (2) LLM inference. Execution length factors in parallel execution. For symbolic domains, we estimate parallel plan length using a dynamic programming check for agent conflicts (optimal for 2 agents, heuristic for more). For embodied domains, we count environment steps, where agents act in parallel but may coordinate subgoals by waiting for other agents.

Overall Results. We show results for SA PDDL, MA PDDL, and TwoStep across five domains with 20 problems each, for 1–4 agents, in Figure 1. We observe that TwoStep consistently lowers planning time compared to MA PDDL for most domains and typically yields shorter execution lengths in 3/5 domains. In highly sequential tasks like `BLOCKSWORLD`, MA PDDL sometimes finds shorter plans by exhaustively searching the larger state space. In domains lacking strong agent-specific states (e.g., `TYREWORLD`), MA PDDL gains no concurrency advantage. By contrast, TwoStep exploits LLM-driven subgoal decomposition and can reduce planning times by up to 64.7% (for 4 agents) and execution lengths by up to 13.2%, compared to SA PDDL. Moreover, it works with unmodified single-agent domains, benefiting from the LLM's partitioning of subgoals.

In complex domains like `TERMES` multiple agents increase the search space for MA PDDL, while TwoStep divides the task into parallelizable subplans.

We demonstrate that this method generalizes to the embodied domain with improved performance over MA PDDL and through human studies find that LLM subgoals approach performance of human expert generated subgoals.

Overall, these results show that TwoStep leverages parallelization with minimal domain engineering and scales more efficiently than naive multi-agent PDDL for larger N.

IV. Conclusion

We propose TwoStep, a method to decompose a single agent planning problem into a multi-agent planning problem in several symbolic domains and one embodied domain. TwoStep leverages commonsense from LLMs to effectively divide a problem between any N agents for faster execution, while also preserving execution success using classical planning guarantees. Our results show that LLM-based goal decomposition leads to faster planning time and shorter plan execution steps than the multi-agent PDDL problem on average. We additionally show that LLM-inferred subgoals in TwoStep approximate those specified by a human expert [15].

Acknowledgements

This work was supported by a grant from the Army Research Lab (ARL) Army AI Innovations Institute (A2I2), award number W911NF-23-2-0010.

REFERENCES

[1] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl- and asp-based systems," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, pp. 363–373, 03 2019.

[2] M. Fox and D. Long, "Pddl2.1: An extension to pddl for expressing temporal planning domains," *ArXiv*, vol. abs/1106.4561, 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:1397894

[3] G. Brewka, T. Eiter, and M. Truszczyński, "Answer set programming at a glance," *Commun. ACM*, vol. 54, no. 12, p. 92–103, dec 2011. [Online]. Available: https://doi.org/10.1145/2043174.2043195

[4] V. Lifschitz, "Answer set programming and plan generation," *Artificial Intelligence*, vol. 138, no. 1, pp. 39–54, 2002, knowledge Representation and Logic Programming. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370202001868

[5] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 9118–9147. [Online]. Available: https://proceedings.mlr.press/v162/huang22a.html

[6] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K.-H. Lee, Y. Kuang, S. Jesmonth, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu, "Do as i can, not as i say: Grounding language in robotic affordances," in *6th Annual Conference on Robot Learning*, 2022. [Online]. Available: https://openreview.net/forum?id=bdHkMjBJG_w

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

[8] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500.

[9] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: program generation for situated robot task planning using large language models," *Autonomous Robots*, 08 2023.

[10] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. P. Kaelbling, and M. Katz, "Generalized planning in pddl domains with pretrained large language models," 2023.

[11] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[12] S. S. Kannan, V. L. N. Venkatesh, and B.-C. Min, "Smart-llm: Smart multi-agent robot task planning using large language models," *ArXiv*, vol. abs/2309.10062, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:262055166

[13] Z. Mandi, S. Jain, and S. Song, "Roco: Dialectic multi-robot collaboration with large language models," 2023.

[14] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv*, 2017.

[15] D. Bai, I. Singh, D. Traum, and J. Thomason, "Twostep: Multi-agent task planning using classical planners and large language models," 2025. [Online]. Available: https://arxiv.org/abs/2403.17246