HiTAMP: A Hierarchical LLM-Modulo Planner for Feasible Long-Horizon Task and Motion Planning

Claudius Kienle[1], Benjamin Alt[2], Oleg Arenz[1] and Jan Peters[1]

I. INTRODUCTION

As robots become more integrated into complex environments for advanced manipulation tasks, their ability to reason over extended action sequences is crucial [1], [2]. This necessitates long-horizon task and motion planning, where traditionally formal languages are being used to define the problem and domain in structured definitions, enabling AI planners to compute a valid task sequence [3], [4]. However, constructing accurate formal descriptions remains a significant challenge, particularly due to the complexity of accurately modeling real-world interactions [5], [6].

Recent advancements leverage Large Language Models (LLMs) as either translators [6]–[10] or planners [5], [11]–[16] to facilitate task planning directly from natural language, removing the need to manually define structured definitions. While promising, these approaches sacrifice the strong feasibility guarantees provided by classical AI planners and often struggle with reliably generating correct long-horizon plans [5], [6], [17]. To address these limitations, prior research has proposed verification and correction mechanisms for LLM-generated plans [11], [13], [18], such as selecting the most feasible plan among multiple candidates [5], [9], [11], [18] or iterative re-prompting for refinement [7], [19]. However, these methods often use sparse or no feedback about feasibility violations, forcing the LLM to make educated guesses about errors [18], [19] and require global re-planning [19], increasing computational cost as task complexity grows.

In response, we propose HiTAMP, a novel LLM-augmented hierarchical task and motion planner that generates long-horizon plans from natural language instructions while ensuring plan feasibility (see Fig. 1). Our approach decomposes the planning hierarchically: it first constructs an abstract domain and high-level plan, which is then incrementally refined into sub-plans. This structure enables partial re-planning while maintaining state alignment across hierarchy levels. Additionally, we incorporate motion-level feasibility verification to ground the LLM's world knowledge in observed state transitions from simulation. Finally, we introduce a central reasoner that addresses feasibility violations at the hierarchical level the violation occurs and triggers re-planning only within the affected sub-plan, avoiding unnecessary changes to the rest of the plan.

This work was supported by the German Federal Ministry of Education and Research (project RobInTime, grant 01IS25002B).
[1]Intelligent Autonomous Systems Group, TU Darmstadt, Germany
[2]AICOR Institute for Artificial Inteligence, University of Bremen, Germany

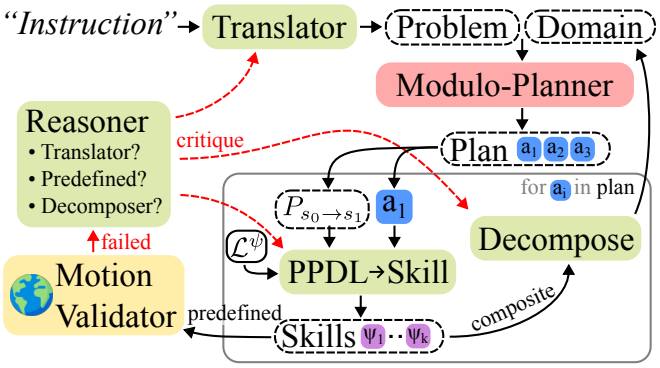

Fig. 1. HiTAMP: The task instruction is translated into high-level PDDL definitions, from which a Modulo-Planner generates an action sequence $a_1, \ldots, a_k$. Each action a_i and its state transition $P_{s_0 \to s_1}$ is either mapped to a predefined robot skill $\psi_i \in \mathcal{L}^\psi$, or decomposed into a sub-plan. Motion verification of each skill ψ_i detects deviations from simulation, sending violations to a central reasoner to identify the cause and trigger re-planning.

II. HIERARCHICAL AI PLANNING

The goal of task planning is to find a feasible sequence of predefined skills $\psi_1, \ldots, \psi_k, \psi_i \in \mathcal{L}^\psi$ that satisfies a natural language instruction i. Given i and initial state s_1, we prompt a LLM to generate a Planning Domain Definition Language (PDDL) domain definition d^1 and problem definition p^1. Rather than enforcing a specific action format or abstraction level, we guide the the LLM to define a small number of high-level actions a_j^1, which are subsequently decomposed. This approach significantly simplifies domain generation and improves correctness. An LLM-modulo planner [11] then generates an action sequence $a_1^1, \ldots, a_n^1$ and automatically corrects any syntactic or semantic errors in the PDDL definitions [11]. To handle high-level actions that may not directly correspond to predefined skills, we introduce a hierarchical decomposition algorithm. For each action a_j^1 in the plan, the algorithm first attempts to map it to a predefined skill $\psi_j \in \mathcal{L}^\psi$ through a process we refer to as *Mapping and Translation*. If the action cannot be directly grounded in the skill library $\mathcal{L}^\psi$, the algorithm triggers *Decomposition*, generating a lower-level sub-plan $a_1^2, \ldots a_m^2$ that implements the intended behavior of the high-level action.

a) Mapping and Translation: Given a domain d, problem p and an action sequence $a_1, \ldots, a_k$, the planner processes actions in a depth-first manner, starting with the first action a_1. To determine whether a_1 corresponds to a predefined skill, we prompt the LLM with the skill library $\mathcal{L}^\psi$ and the PDDL definition of a_1. Based on that, the model proposes a sequence of predefined skills that could

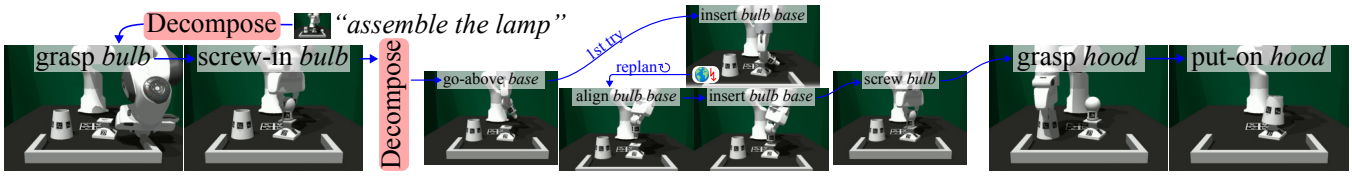

Fig. 2. Planning a long-horizon task with the hierarchical planner from natural language, including the decomposition of high-level actions such as *screw-in* and re-planning due to a motion verification violation of the *insert bulb base* skill.

implement the action. If this sequence consists of a single skill, the action is marked as a leaf node and passed directly to motion-level feasibility verification. If multiple skills are required, the action is passed on for *Decomposition*.

b) Decomposition: For actions that cannot be directly mapped to a single predefined skill, we apply hierarchical decomposition. The process splits a high-level action a_i^1 into a sequence of lower-level actions $a_1^2, \ldots, a_m^2$, e.g. decomposing *Grasp* into *Approach*, *CloseGripper*, and *Lift*. To do this, we define a new problem definition $P^1_{s_i \to s_{i+1}}$ that captures the state before and after executing a_i^1. The LLM is then prompted to generate a new domain that defines lower-level actions required to achieve the same state transition.

III. MOTION VERIFICATION AND RETRACTION

Since all PDDL actions are generated by the LLM, their preconditions and effects are not guaranteed to be correct. To address this, we developed a motion verification process that checks the correctness of PDDL actions in simulation. After planning a leaf action a_i, that action is executed in simulation and its effects are verified against those observed in simulation. If motion verification fails, the system identifies the most likely point in the planning hierarchy where the discrepancy between the LLM's world knowledge and the simulation occurred. A centralized error reasoner (itself a correspondingly prompted LLM) processes the entire chat history along with the motion verification results to pinpoint the step at which the error likely originated. Once the error location is identified, the system retracts to that point and re-prompts the LLM with the observed discrepancy, realigning the world knowledge with the simulation results.

IV. HIERARCHICAL KNOWLEDGE CONSISTENCY AND PLAN ALIGNMENT

Dividing the planning problem hierarchically presents two key challenges: preserving knowledge across abstraction levels and ensuring alignment between planning problems at different hierarchy levels to ultimately produce a coherent hierarchical plan. To maintain knowledge consistency, PDDL predicates and object types are retained across hierarchy levels and made available for decomposition. This enables the LLM to utilize these predicates when defining PDDL actions, which is crucial, as the motion verification relies on the predefined predicates to verify action feasibility. Additionally, LLMs at lower hierarchy levels have access to the chat history from higher levels, ensuring that information generated by upper-level LLMs remains available. Our observations indicate that omitting this history by encapsulating hierarchy levels and their LLM calls leads to reduced robustness, redundant decompositions, and instability in planning.

When an action a^1 with effects e^1 is decomposed into sub-actions $a_1^2, \ldots, a_m^2$, the effects of a^1 must correspond to, or *align with*, the combined effects of a_1^2 through a_m^2. While sub-actions cannot omit any of the effects from the original action, they may introduce additional effects not defined in the initial problem. For example, if the action 'grasp' has the effect `grasped object`, but the decomposition adds `closed-gripper`, it leads to an unintended expansion of effects. Such expansions, where additional predicates are modified beyond those specified in the original action, can disrupt the alignment between the hierarchy levels. To address this, we re-prompt the decomposition model to correct the effects of a^1 and realign them with the sub-actions, followed by re-planning as necessary.

V. EXPERIMENTS

We evaluate HiTAMP on the FurnitureBench Benchmark [20], which provides complex long-horizon tasks in a simulated environment. Given a skill library and predicates to evaluate the state of the simulation, we task the planner to assemble a lamp, as shown in Figure 2. The hierarchical decomposition reduces complexity by first generating a high-level action sequence consisting of three distinct actions. During the initial decomposition of the *screw-in* action, the lower-level action sequence misses a necessary *align* skill, causing the insertion to fail. However, through *motion verification*, HiTAMP detects a deviation of the expected state with the state observed in simulation. This triggers re-planning, which corrects the decomposition by adding a new action to align the parts, leading to successful insertion and screwing of the bulb. Additional experiments on the remainder of FurnitureBench, planner benchmarks, ablation studies as well as real-world robot experiments are forthcoming.

VI. CONCLUSION

We introduce HiTAMP, a hierarchical task and motion planner designed to solve long-horizon tasks via the generation of reliable, simulation-verified plans. By decomposing tasks hierarchically, our approach reduces the complexity of generating correct domain definitions with LLMs and enables the reuse of previously decomposed skills. Each planned skill is grounded in simulation, aligning the LLM's world knowledge with observations. We further introduce an centralized reasoner that adapts and realigns plans based on simulation feedback, resulting in more reliable and feasible planning.

REFERENCES

[1] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A Survey of Multi-Objective Sequential Decision-Making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, Oct. 2013. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10836

[2] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 239–252, 2022, publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9868234/

[3] M. Ghalleb, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, *et al.*, "PDDL - The Planning Domain Definition Language," *Technical Report, Tech. Rep.*, 1998. [Online]. Available: https://www.researchgate.net/profile/Craig-Knoblock/publication/2278933_PDDL-_The_Planning-Domain_Definition_Language/links/0912f50c0c99385e19000000/PDDL-The-Planning-Domain-Definition-Language.pdf

[4] M. Fox and D. Long, "PDDL2. 1: An extension to PDDL for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10352

[5] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2Motion: from natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, Dec. 2023. [Online]. Available: https://link.springer.com/10.1007/s10514-023-10131-7

[6] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati, "On the Planning Abilities of Large Language Models - A Critical Investigation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 75 993–76 005, Dec. 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/efb2072a358cefb75886a315a6fcf880-Abstract-Conference.html

[7] E. Gestrin, M. Kuhlmann, and J. Seipp, "NL2Plan: Robust LLM-Driven Planning from Minimal Text Descriptions," May 2024, arXiv:2405.04215 [cs]. [Online]. Available: http://arxiv.org/abs/2405.04215

[8] 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," Sept. 2023, arXiv:2304.11477. [Online]. Available: http://arxiv.org/abs/2304.11477

[9] S. Mahdavi, R. Aoki, K. Tang, and Y. Cao, "Leveraging Environment Interaction for Automated PDDL Generation and Planning with Large Language Models," July 2024, arXiv:2407.12979. [Online]. Available: http://arxiv.org/abs/2407.12979

[10] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati, "Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning," Nov. 2023, arXiv:2305.14909. [Online]. Available: http://arxiv.org/abs/2305.14909

[11] A. Gundawar, K. Valmeekam, M. Verma, and S. Kambhampati, "Robust Planning with Compound LLM Architectures: An LLM-Modulo Approach," Nov. 2024, arXiv:2411.14484 [cs]. [Online]. Available: http://arxiv.org/abs/2411.14484

[12] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, *et al.*, "Code as Policies: Language Model Programs for Embodied Control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 9493–9500. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10160591?casa_token=WFKbBaJAs6IAAAAA:KPr_ULH6fkAWMMyLaS01pZ2_xkQEajIgZyrD6wkN1jKE-wfvtX3DOwk8Gmb26BqUCzNuS_gLgQ

[13] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, "LLM3:Large Language Model-based Task and Motion Planning with Motion Failure Reasoning," Aug. 2024, arXiv:2403.11552. [Online]. Available: http://arxiv.org/abs/2403.11552

[14] S. S. Kannan, V. L. N. Venkatesh, and B.-C. Min, "SMART-LLM: Smart Multi-Agent Robot Task Planning using Large Language Models," Mar. 2024, arXiv:2309.10062. [Online]. Available: http://arxiv.org/abs/2309.10062

[15] H. Fan, X. Liu, J. Y. H. Fuh, W. F. Lu, and B. Li, "Embodied intelligence in manufacturing: leveraging large language models for autonomous industrial robotics," *Journal of Intelligent Manufacturing*, Jan. 2024. [Online]. Available: https://doi.org/10.1007/s10845-023-02294-y

[16] S. Kambhampati, "Can Large Language Models Reason and Plan?" *Annals of the New York Academy of Sciences*, p. nyas.15125, Mar. 2024, arXiv:2403.04121 [cs]. [Online]. Available: http://arxiv.org/abs/2403.04121

[17] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati, "Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change)," Nov. 2022. [Online]. Available: https://openreview.net/forum?id=wUU-7XTL5XO

[18] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "AutoTAMP: Autoregressive Task and Motion Planning with LLMs as Translators and Checkers," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 6695–6702. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10611163

[19] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma, "ISR-LLM: Iterative Self-Refined Large Language Model for Long-Horizon Sequential Task Planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 2081–2088. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10610065

[20] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "FurnitureBench: Reproducible real-world benchmark for long-horizon complex manipulation," *The International Journal of Robotics Research*, p. 02783649241304789, Feb. 2025, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/02783649241304789