A Roadmap to Guide the Integration of LLMs in Hierarchical Planning

Israel Puerta-Merino, Carlos Núñez-Molina, Pablo Mesejo, Juan Fernández-Olivares

University of Granada, Spain
Andalusian Institute of Data Science and Computational Intelligence (DaSCI)
israelpm01@ugr.es, ccaarlos@ugr.es, pmesejo@go.ugr.es, faro@decsai.ugr.es

Abstract

Recent advances in Large Language Models (LLMs) are fostering their integration into several reasoning-related fields, including Automated Planning (AP). However, their integration into Hierarchical Planning (HP), a subfield of AP that leverages hierarchical knowledge to enhance planning performance, remains largely unexplored. In this preliminary work, we propose a roadmap to address this gap and harness the potential of LLMs for HP. To this end, we present a taxonomy of integration methods, exploring how LLMs can be utilized within the HP life cycle. Additionally, we provide a benchmark with a standardized dataset for evaluating the performance of future LLM-based HP approaches, and present initial results for a state-of-the-art HP planner and *LLM planner*. As expected, the latter exhibits limited performance (3% correct plans, and none with a correct hierarchical decomposition) but serves as a valuable baseline for future approaches.

Introduction

Hierarchical Planning (HP) is a subfield within Automated Planning (AP) comprising planning methods that incorporate hierarchical knowledge. This hierarchical knowledge can be leveraged to speed up planning and, also, to integrate human expert problem-solving knowledge. While Large Language Models (LLMs) are being gradually integrated in various AI domains, including AP, their application to HP remains underexplored, with only a few studies tangentially addressing this topic (Yang, Zhang, and Hou 2024; Dai et al. 2024; Tse 2024; Song et al. 2023). Our contribution within this work, therefore, is a roadmap to fill this gap, exploring the potential of LLMs for HP.

We have analyzed existing literature on AP with LLMs, as well as current reviews (Pallagani et al. 2024; Huang et al. 2024; Valmeekam et al. 2022). Observing that most methods in AP that are similarly applicable to HP, we have classified these methods, building a taxonomy to bring HP closer to the existing LLM integration techniques. Our taxonomy classifies the integration methods according to two different dimensions: the *Planning Process Role* (in which part of the HP life cycle is the LLM applied – i.e. problem definition, plan elaboration or post-processing) and the *LLM Improvement Strategy* (which LLM-based approach are used to improve the LLM performance – i.e. giving extra knowledge or making multiple calls). Given that HP is a subset of AP, this classification is broadly applicable to AP as well.

To facilitate evaluation and comparisons among methods, we propose a standardized dataset and benchmarking framework based on the 2023 International Planning Competition (IPC-2023) HTN tracks, the most recent competition for HP solvers. Specifically, we suggest using the total-order track of the IPC-2023 dataset as a benchmark dataset.[1] As a baseline, we implement and evaluate a basic LLM Planner (direct planning using an LLM without any improvement strategies), the simplest method of our taxonomy, on this dataset. We use Llama-3.1-Nemotron-70B-Instruct (Wang et al. 2024), one of the highest-performing LLMs available, and the model that we plan to use in subsequent experiments. We also provide the results from PandaDealer-agile-lama (Olz, Höller, and Bercher 2023), the IPC-2023 Total-Order Satisficing track winner and state-of-the-art HP solver.

In summary, this work offers a roadmap to guide the future research in the integration of LLMs and HP, which is provided through two main contributions: the taxonomy of LLM in HP integration methods, illustrating the magnitude of this field and revealing how much work remains to be done; and the proposed benchmark, providing a tool for the evaluation and comparison of developed and future methods. We hope that this roadmap will inspire and guide future research in this promising yet underexplored field.

Taxonomy

To bridge the gap between HP and existing LLM Integration techniques, and to explore this expansive field, we propose a taxonomic framework that highlights the identified methods. This classification mainly draws from AP's state-of-the-art, so many of these methods also apply to AP. To provide context for the subsequent sections, we include illustrative examples of remarkable AP Integration methods from current literature. Each of the methods within this taxonomy represents a subset of techniques rather than a precise implementation, and may encompass multiple approaches. Furthermore, these methods are non-exclusive, meaning that planning agents can be designed to explore multiple combinations, making this a huge field to explore. This classification is intended to be a starting point, not a fixed framework, with room for exploration and refinement.

[1] Website of the IPC-2023: https://ipc2023-htn.github.io

Our proposed taxonomy is structured along two dimensions: the *Planning Process Role*, categorizing the stages of the HP life cycle where an LLM can be applied: problem definition, plan elaboration and post-processing (i.e., including plan translation and explanation to final user). And *LLM Improvement Strategy*, which encompasses general strategies used to enhance LLM performance, which are applicable regardless of the role they are used for: knowledge enhancement and multiple calls. In this last dimension, despite several LLM Reasoning Strategies have been studied, we have focused our classification on the main different strategies that, we have observed, are commonly used in AP, as we consider that are the most interesting techniques to initially explore and evaluate in HP. These strategies mainly consist on using extra knowledge or augmenting the number of LLM executions.

Planning Process Role

LLMs can assume various roles across the three general steps of a planning process: (1) problem definition, (2) plan elaboration, and (3) post-processing. Each step is further divided into the distinctive LLM integration methods observed in the literature. Table 1 summarizes this classification.

Problem Definition. An HP Problem includes the same elements as an AP problem (*Actions, Initial State* and *Goal*), along with the high-level actions (*Tasks*) that represents the hierarchical information about the environment. Each element can be generated using an LLM through two main approaches: *Translation*, when all necessary information is explicitly and previously provided and we only want the LLM to restructure the provided information into a target format (Liu et al. 2023); and *Generation*, when the information is partially or implicitly provided, so the LLM is inferring or assuming missing information based on reasoning (Gestrin, Kuhlmann, and Seipp 2024).

Plan Elaboration We categorize this group based on the role of the LLM in the problem solving process:

- **LLM Planner**. In this basic setup, the LLM itself functions as the planner (Silver et al. 2022). While LLM improvement strategies can be applied, there is not external planner or explicit search process here, so the LLM the model is responsible for the entire planning process.

- **Graph Search**. The LLM is embedded in the planner, which is done within an explicit search algorithm, where the LLM can perform one or more roles, like *Node Expansion* (generating the next possible actions), *Selection* (choosing the next action), *Heuristic provision* (scoring states), *Model Elicitation, Backtracking, Aggregation and Pruning*. Some remarkable architectures are RAP (Hao et al. 2023), integrating an LLM into a MCTS Planner; GoT (Besta et al. 2024), implementing an LLM in most of the mentioned roles to solve reasoning problems; and SayCan (Ahn et al. 2022), which utilizes an LLM as a probabilistic relevance (heuristic) scorer.

Problem Definition	Generation	*Actions* *Tasks* *Initial State* *Goal*
	Translation	*Actions* *Tasks* *Initial State* *Goal*
Plan Elaboration	*LLM Planner*	
	Graph Search	*Expansion* *Selection* *Elicitation* *Backtracking* *Heuristic* *Aggregation* *Pruning*
	Guidance	*Preferences* *Initialization*
Post-Processing	*Translating the Plan* *Explaining the Plan*	

Table 1: Summary of the possible roles that an LLM can perform during the HP life cycle. This classification is detailed in the *Planning Process Role* section.

- **Planning Guidance**. Here, the LLM is external to the planner, but assists it by providing assistance. This guidance can be an *initial plan* that the planner can refine (Valmeekam et al. 2023) or environment *preferences* to narrow the search space (Sharan et al. 2023).

Post-Processing Once a plan is developed, an LLM can be used to refine it, by *Translating the plan* into another data structure or language, typically an executable format (e.g. for a robot to run it) or natural language (Liu et al. 2023). Or *Explaining the Plan*, where de LLM has to generate more detailed information, commonly in natural language, based on the provided plan (Simon and Muise 2022).

LLM Improvement Strategies

When performing any task using an LLM, it can be directly executed to obtain an output. However, their results are often suboptimal. To address this, there are some strategies to enhance the LLM performance which have been used in several reasoning-related fields, as math, question answering or AP. Building on this foundation, we have identified the strategies commonly used in AP, and extended them to the specific case of HP. In this section, we propose a classification of strategies that can be employed in HP to improve the LLM performance, regardless of the role they are used for.

LLM performance can be enhanced by either providing more knowledge about the problem or increasing the number of LLM calls used during problem-solving. Each of these approaches is also categorized into the distinctive strategies. This classification is detailed along next subsections and summarized in Table 2. Note that these strategies are neither mandatory nor mutually exclusive, meaning that it is

possible to use none, some, or all of them simultaneously.

Knowledge Enhancement We can split the LLM knowledge enhancement strategies into two main perspectives: with previous knowledge (providing additional information relevant to the problem before starting to solve it) or through feedback (the model is iteratively provided with extra information based on its previous outputs, while solving the task). Both perspectives, moreover, may have different approaches depending on the location where the knowledge is applied:

- **Previous**. This approach involves providing extra information to the LLM before its execution. *Fine-tuning* is the traditional method in the field of deep learning, which involves adjusting the model's internal weights with supplementary data (Pallagani et al. 2022). Alternatively, knowledge can be directly provided through the model's prompt, which is a more flexible and accessible option than fine-tuning, as it does not require a training process. Providing extra information is achieved through input-output examples (shots) that enhance generated responses (Song et al. 2023). If the shots provide useful environmental information (e.g., domain information) this is called *in-context prompting*; if they only aim to elicit an specific output structure, it is referred to as *out-of-context prompting*. Furthermore, we can also provide knowledge about how to reason, about the problem itself. *Chain of Thoughts* (CoT) (Wei et al. 2022) is the best example of this, a different prompting strategy that encourages the LLM to generate a reasoning process before providing the final answer. This is typically achieved by using *few-shot prompting* with reasoning examples, though reasoning can also be elicited without examples, known as *Zero-shot CoT* (Kojima et al. 2022).
- **Feedback.** Unlike the previous paradigm, here the extra information is provided during the task solving process: rather than simply accepting the output of an LLM execution, we can iteratively improve the result of the LLM by providing it feedback based on its previous outputs. This feedback can come from various sources (humans, the environment, an external module, another LLM, etc.). *Reinforcement Learning* is the classical method of learning from feedback in machine learning, where the knowledge is applied by modifying the model's internal weights, based on a reward score (Yao et al. 2020). Alternatively, feedback can be used to dynamically adjust and improve the prompt (*Prompt Correction*), where prominent examples in the literature are Self-Refine (Madaan et al. 2024) and Reflexion (Shinn et al. 2024). Finally, the model's learning can also be represented through an external *Memory Module*, being Voyager a notable architecture that uses this approach (Wang et al. 2023).

Multiple Calls To improve the LLM's performance through multiple calls, there are two main perspectives: the problem can be divided into simpler sub-problems that the LLM solve on each call (Decomposition); or the LLM solves the whole problem several times and we take advantage of the varied information generated by the multiple outputs (Revision). Both perspectives can also be addressed through two different approaches, depending on whether an output is used for the next call or not (*Sequential* or *Parallel* calls).

Knowledge	Previous	*Fine-tuning* *Prompting* *Chain of Thoughts*
	Feedback	*Reinforcement* *Prompt Correction* *Memory*
Multi-Calls	Decomposition	*Sequential Calls* *Parallel Calls*
	Revision	*Sequential Calls* *Parallel Calls*

Table 2: Summary of the possible strategies that can be followed to improve the LLM's performance within the HP life cycle, regardless of the specific role. This classification is detailed in the *LLM Improvement Strategies* section.

- **Decomposition.** A problem can be sequentially decomposed: each LLM call's output generates an intermediate step that is utilized for the next call's input. For instance, to generate a plan, we could iteratively ask the LLM to only generate the next action of a partial plan, than attempting to generate the entire plan in a single call (Huang et al. 2022). Otherwise, the problem can be decomposed in parallel: each LLM call can returns a list simpler sub-problems which can be independently solved (i.e. the classical Divide and Conquer strategy). This is illustratively used in Generative Agents (Park et al. 2023).
- **Revision.** We can make the multiple calls to sequentially refine the LLM's output: we ask the LLM to initially give us a complete but simple (general) solution that iteratively becomes more detailed through the calls (Liu et al. 2024). Otherwise, we can do a parallel approach by asking the LLM to entirely solve the problem multiple times and, then, combine the different results into a final output. Self-consistency (Wang et al. 2022) is a prominent approach, which combines the several outputs by selecting the most common response, but various criteria can be used to choose the final output.

Benchmark

Hierarchical Task Networks (HTN) Planning (Nau et al. 2003) is the most widely used and studied approach within the HP field, making it a suitable reference point for experimenting with and evaluating various LLM integration methods. We propose using the total-order track from IPC-2023, which contains 22 domains, each with dozens of problems. A detailed breakdown of this information, along with a summary of the experimental results, is provided in Table 3.

To assess and compare the performance of future implementations, we establish two reference points. First, we consider the winner of the IPC-2023 Total-Order Satisficing track, PandaDealer-agile-lama (Olz, Höller, and Bercher 2023) as an upper bound, as it represents the current state of the art in HP solvers. Second, we include as baseline a basic LLM Planner (i.e. using an LLM directly to plan without

providing any LLM Improvement Strategy) representing a lower bound for LLM Integration. This approach, being the simplest method in our taxonomy, serves as a foundational point of comparison and a starting point for the roadmap.

Execution and Experimentation Considerations

The PandaDealer score within the dataset have been sourced from the published IPC-2023 results. This score represents the ratio C^*/C between the cost of a reference plan (C^*) and the best obtained plan (C). Due to the unavailability of these reference plans, we could not compute the same score for the LLM Planner. Nonetheless, this score serves as a reliable reference for assessing the quality of PandaDealer's performance. For generative plans, however, additional properties need to be evaluated, such as semantic coherence within the plan (which does not need to be measured in symbolic planners). To address this, we adopted alternative metrics more suitable for analyzing the quality of generative plans: *Plan Feasibility* (the plan is syntactically correct), *Plan Correctness* (it is executable and reaches a goal state), *Decomposition Feasibility* (the hierarchical decomposition of the plan is syntactically correct) and *Decomposition Correctness* (the decomposition aligns with the generated plan).

We utilize Llama-3.1-Nemotron-70B-Instruct, one of the highest-performing LLMs available (Wang et al. 2024), as the LLM Planner. We utilize the official IPC verifier to score the generated plans along the different proposed metrics. The execution and validation source codes, as well as the generated plans, and obtained results are allocated on GitHub. URL: https://github.com/Corkiray/HTN-LLM.

Results Discussions

In this preliminary work, we have obtained the results for 15 out of the 23 domains in the dataset, not the complete set, due to temporal and computational limitations. Nevertheless, these results are sufficient to illustrate the notably low performance of the LLM Planner, which is expected given the simplicity of the method and the absence of any LLM Improvement Strategies. As shown in Table 3, the LLM Planner fails to generate feasible plans in nearly 70% of the problems, highlighting the LLM's limited ability to interpret and adhere to a specific format. Even more remarkable is the proportion of correct plans: with only 4% (i.e. 13% of the feasible plans), it shows the difficulty that a basic LLM encounters in planning correctly. A similar trend is observed in the number of feasible decompositions which, with only 3% correct, indicates a significant performance drop in handling a specific format with increasing complexity. Interestingly, correct plans and feasible decompositions are often disjoint (i.e. the LLM achieves either one result or the other, but not both). Consequentially, the LLM has been unable to produce a correct hierarchical decomposition for any problem. This observation is explainable considering the LLM's inherent architecture as a transformer, which operates within a static computational capacity. An LLM cannot dynamically adjust its processing time based on problem complexity, and as a result, its ability to address multiple demanding requirements simultaneously (i.e. plan correctness and decomposition feasibility) is limited.

		Panda	**LLM Planner**			
Domain	N	Score	FP	CP	FD	CD
Assembly	30	0.89	15	3	0	0
Barman	20	0.78	4	1	0	0
Blocks	30	0.77	9	0	1	0
Depots	30	0.90	4	0	0	0
Factories	20	0.67	6	0	0	0
Freecell	60	0.13	38	18	0	0
Hiking	30	0.83	0	0	0	0
lamps	30	0.48	5	1	1	0
Logistics	80	0.98	14	1	1	0
Multiarm	74	0.95	30	0	2	0
Robot	20	0.92	13	1	5	0
Satellite	20	0.92	11	0	2	0
Towers	20	0.65	5	0	0	0
Transport	40	0.73	10	0	4	0
Woodwork	30	0.69	26	0	0	0
Total	614	14.42	190	25	16	0
Average:		**0.80**	**0.31**	**0.04**	**0.03**	**0**

Table 3: Summary of the information provided by the benchmark, listing the number of problems in each domain of the dataset and the performance reported by each planner. The score of PandaDealer is the one provided by the IPC-2023 results. The scores utilized for the LLM Planner are explained in the *Benchmark* section, representing the number of feasible plans (FP), correct plans (CP), feasible decompositions (FD) and correct decompositions (CD).

Conclusion

In this work, we propose a roadmap to guide future research in the integration of LLMs and HP, as this remains a largely unexplored field. This roadmap is centered on two key contributions: a taxonomy and a benchmark. The taxonomy categorizes the main integration methods, structured along two dimensions: the first one considers the various roles that an LLM could fulfill within the HP life cycle. The second one focuses on strategies to enhance LLM performance, irrespective of the specific role in which they are applied. The benchmark introduces a dataset and provides initial results that serves as reference in subsequent experimentation. These results include the performance of a state-of-the-art HP solver and an LLM Planner using one of the best-performing LLMs available, but without leveraging any improvement strategy. As expected, the results reveal the LLM's limited performance in solving HP problems, but they establishes a baseline to evaluate succeeding improvements. Promising future directions include exploring the planning capabilities of an LLM Planner augmented with improvement Strategies to overcome the limitations identified in this study. Furthermore, exploring integration in additional aspects of the HP life cycle, such as Plan Monitoring or Exception Management, is beneficial to expand the boundaries of the proposed taxonomy. Finally, developing new architectures within the outlined Planning Process Roles offers a pathway to systematically investigate and advance this unexplored field.

References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI*.

Dai, Z.; Asgharivaskasi, A.; Duong, T.; Lin, S.; Tzes, M.-E.; Pappas, G.; and Atanasov, N. 2024. Optimal scene graph planning with large language model guidance. In *Prodeeding of the ICRA*.

Gestrin, E.; Kuhlmann, M.; and Seipp, J. 2024. NL2Plan: Robust LLM-Driven Planning from Minimal Text Descriptions. *arXiv preprint arXiv:2405.04215*.

Hao, S.; Gu, Y.; Ma, H.; Hong, J. J.; Wang, Z.; Wang, D. Z.; and Hu, Z. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the ICML*.

Huang, X.; Liu, W.; Chen, X.; Wang, X.; Wang, H.; Lian, D.; Wang, Y.; Tang, R.; and Chen, E. 2024. Understanding the planning of LLM agents: A survey. *arXiv preprint arXiv:2402.02716*.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. In *Proceedings of the NeurIPS*.

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

Liu, Y.; Palmieri, L.; Koch, S.; Georgievski, I.; and Aiello, M. 2024. Delta: Decomposed efficient long-term robot task planning using large language models. *arXiv preprint arXiv:2404.03275*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2024. Self-refine: Iterative refinement with self-feedback. In *Proceedings of the NeurIPS*.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of artificial intelligence research*.

Olz, C.; Höller, D.; and Bercher, P. 2023. The PANDADealer System for Totally Ordered HTN Planning. *IPC HTN Tracks*.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Horesh, L.; Srivastava, B.; Fabiano, F.; and Loreggia, A. 2022. Plansformer: Generating symbolic plans using transformers. *arXiv preprint arXiv:2212.08681*.

Pallagani, V.; Muppasani, B. C.; Roy, K.; Fabiano, F.; Loreggia, A.; Murugesan, K.; Srivastava, B.; Rossi, F.; Horesh, L.; and Sheth, A. 2024. On the prospects of incorporating large language models (llms) in automated planning and scheduling (aps). In *Proceedings of the ICAPS*.

Park, J. S.; O'Brien, J.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the UIST*.

Sharan, S.; Pittaluga, F.; Chandraker, M.; et al. 2023. Llm-assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2024. Reflexion: Language agents with verbal reinforcement learning. In *Proceedings of the NeurIPS*.

Silver, T.; Hariprasad, V.; Shuttleworth, R. S.; Kumar, N.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. PDDL planning with pretrained large language models. In *Proceedings of the NeurIPS*.

Simon, N.; and Muise, C. 2022. TattleTale: storytelling with planning and large language models. In *Proceedings of the ICAPS SPARK*.

Song, C. H.; Wu, J.; Washington, C.; Sadler, B. M.; Chao, W.-L.; and Su, Y. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the ICCV*.

Tse, C. 2024. Improving Human-Robot Communication of Hierarchical Task Planning through LLMs. *Brown University*.

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2022. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *Proceedings of the NeurIPS*.

Valmeekam, K.; Sreedharan, S.; Marquez, M.; Olmo, A.; and Kambhampati, S. 2023. On the planning abilities of large language models (a critical investigation with a proposed benchmark). *arXiv preprint arXiv:2302.06706*.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, Z.; Bukharin, A.; Delalleau, O.; Egert, D.; Shen, G.; Zeng, J.; Kuchaiev, O.; and Dong, Y. 2024. HelpSteer2-Preference: Complementing Ratings with Preferences. *arXiv preprint arXiv:2410.01257*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the NeurIPS*.

Yang, R.; Zhang, F.; and Hou, M. 2024. OceanPlan: Hierarchical Planning and Replanning for Natural Language AUV Piloting in Large-scale Unexplored Ocean Environments. *arXiv preprint arXiv:2403.15369*.

Yao, S.; Rao, R.; Hausknecht, M.; and Narasimhan, K. 2020. Keep calm and explore: Language models for action generation in text-based games. *arXiv preprint arXiv:2010.02903*.