On The Planning Abilities of OpenAI's o1 Models: Feasibility, Optimality, and Generalizability

**Kevin Wang[∇], Junbo Li[∇], Neel P. Bhatt[∇], Yihan Xi,
Qiang Liu, Ufuk Topcu, and Zhangyang Wang[†] [*]**

Abstract

Recent advancements in Large Language Models (LLMs) have showcased their ability to perform complex reasoning tasks, but their effectiveness in planning remains underexplored. In this study, we evaluate the planning capabilities of OpenAI's o1 models across a variety of benchmark tasks, focusing on three key aspects: feasibility, optimality, and generalizability. Through empirical evaluations on constraint-heavy tasks and spatially complex environments, we highlight o1-preview's strengths in self-evaluation and constraint-following, while also identifying bottlenecks in decision-making and memory management, particularly in tasks requiring robust spatial reasoning. Our results reveal that while o1-preview outperforms GPT-4, the model often generates suboptimal solutions with redundant actions and struggles to generalize effectively in spatially complex tasks. This pilot study provides foundational insights into the planning limitations of LLMs, offering key directions for future research on improving memory management, decision-making, and generalization in LLM-based planning.

1 Introduction

Large Language Models (LLMs) have significantly changed the landscape of artificial intelligence, achieving impressive results in various language-related tasks, such as chatbots, math, and coding, etc. One of the areas, that remains yet to be fully claimed by LLMs, is the use of language agents for planning in the interactive physical world [Huang et al., 2022a,b, Singh et al., 2023, Lin et al., 2023]. Previous scrutiny [Liu et al., 2023, Valmeekam et al., 2024a, 2023, 2024b] pointed out that despite advanced inference reasoning techniques like Chain-of-Thought (CoT) [Wei et al., 2022] and Tree-of-Thought (ToT) [Yao et al., 2024], LLMs still struggle in making success plans without relying on external tools, such as a PDDL planner [Liu et al., 2023, Lyu et al., 2023].

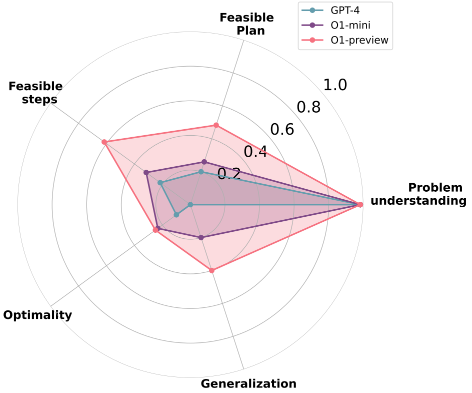


Figure 1: Overall comparison of GPT-4, o1-mini,and o1-preview, on key planning perspectives defined by us.

The recent release of OpenAI's o1 models [OpenAI, 2024], trained with reinforcement learning to naturally employ CoT reasoning, have reached new heights in problem-solving, particularly in mathematics and code generation. This suggests potential for planning, a seemingly related area. A recent work [Valmeekam et al., 2024a] evaluated the success rates of o1 and other LLMs on

---

[*] All authors are with the University of Texas at Austin, Austin, TX, USA. [∇] Equal contribution and co-first authors; [†] Corresponding author; email: `atlaswang@utexas.edu`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Blocksworld and its variants from PlanBench [Valmeekam et al., 2024b], demonstrating enhanced capabilities, extending the boundaries of what LLMs can accomplish in planning tasks. Despite these improvements, o1 is still far from perfect, raising our curious question: *where does the o1 model still fall short, and how can one systematically identify its limitations?*

Our work takes a deeper dive into the planning process to perform a more detailed analysis across a broader range of planning tasks building on [Valmeekam et al., 2024a]. Rather than merely measuring the plan success rate as previous studies did [Valmeekam et al., 2024a, Liu et al., 2023], we classify the different types of errors LLMs make during their planning. Through extensive empirical evaluations, we analyze o1's performance across various domains from three key perspectives: *plan feasibility, plan optimality*, and *plan generalizability*. We aim to offer a clearer understanding of the limitations of LLMs in the entire planning pipeline, and to facilitate future finer-grained diagnostics.

2 Planning Ability Evaluation: Three Perspectives

We propose evaluating the planning abilities of language model agents from three key perspectives: **feasibility, optimality**, and **generalizability**. By dividing planning abilities into these three perspectives, we ensure a comprehensive evaluation, where each aspect plays a significant role in overall performance. The motivation for this division lies in the varied challenges planning entails, from basic execution to advanced optimization and adaptation across new contexts.

Feasibility Feasibility assesses whether the agent can produce a viable plan to achieve the goal, often referred to as success rate in previous works [Liu et al., 2023]. Feasibility can be further divided into three components:

1. **Ability to create feasible steps** Each step in a plan must be executable within the system, adhering to constraints specific to the problem domain. Constraints might include physical limitations, action order requirements, or other domain-specific rules. In trajectory planning, certain zones may be inaccessible, while in operational planning, tasks may have dependencies that must be respected. We term failures related to this issue as "Inability to Follow Problem Rules" (**IR**). Such errors become more frequent as the complexity or number of rules increases, revealing a need for more sophisticated validation mechanisms.
2. **Ability to generate a feasible plan** Even if individual steps are valid, the overall plan may still fail to achieve the intended goal. The agent might not generate a coherent sequence of actions, leading to dead ends or random exploration. This issue, termed "Inability to Generate a Feasible Plan" (**IP**), grows more prominent in complex tasks requiring advanced reasoning. Models like o1, which demonstrate superior reasoning capabilities, tend to perform better, as they provide more thorough analysis and structured plans.
3. **Ability to understand the problem** Feasibility also hinges on correctly interpreting the problem's initial and goal states. Even with valid steps and an overall plan, misinterpreting the starting conditions or the desired end state can result in errors. Such failures, termed "Misinterpretation of Goal State" (**MG**), are common when plans require deep reasoning over multiple steps.

Optimality While feasibility ensures that a plan can be successfully executed, optimality pertains to how efficiently the plan achieves its goal. In many real-world scenarios, a feasible plan is not enough; the plan must also be resource-efficient, minimizing unnecessary actions, time, and cost. In this context, optimality refers to whether the language agent can generate the most efficient plan, avoiding redundant steps or suboptimal decisions. For example, in a robotic task, an optimal plan would minimize the number of movements or tool changes required to complete the task, whereas a suboptimal plan might include unnecessary repetitions or idle steps. Failures in optimality, termed "Lack of Optimality" (**LO**), arise when a plan, although feasible, includes extraneous or inefficient actions that prevent it from being considered the best solution.

Our experiments suggest that while advanced models like o1-preview show some improvements in generating optimal plans compared to previous models, the gap between feasibility and true optimality remains significant. This highlights the inherent difficulty for language models to reason not only about what needs to be done but also about how to do it in the most efficient manner. Developing strategies to incorporate cost-sensitive reasoning or employing more advanced search and pruning techniques could help mitigate this issue.

Generalizability Generalizability examines whether a language model can successfully plan across a diverse range of scenarios, including those it may not have explicitly encountered during training. Generalization becomes especially challenging when the agent must deal with abstract representations of actions or work in environments where the semantics of actions differ from those in natural language. This capability is crucial for robust performance in real-world applications, where the agent may encounter unfamiliar contexts or have to work with symbolic representations that are not directly tied to the real-world semantics - yet still following consistent logical structures.

Inspired by [Valmeekam et al., 2024b], we test whether the agent can construct valid plans even when actions are represented by arbitrary symbols, devoid of any inherent natural language meaning. In these cases, the agent's ability to generalize reflects its deeper understanding of the underlying structure and logic of planning tasks, independent of the specific linguistic cues it was trained on. This aspect is critical in fields like robotics, where planning often involves symbolic reasoning and manipulation of abstract entities. Our experiments (e.g., Figure 5) indicate that generalization remains a significant challenge for current models, especially in more complex, spatially dynamic environments. Models like o1-preview show a clear degradation in performance when transitioning from familiar tasks to generalized ones, suggesting that their learned representations are often too closely tied to specific task domains.

We present the evaluation of several interactive planning tasks from both feasibility (IR, IP, MG) and optimality perspectives in Table 1. The success rate (SR) is defined as the rate of instances where none of IR, IP, or MG occurred. For detailed results on each task, please refer to Section A.

	GPT-4					o1-mini					o1-preview				
	IR	**IP**	**MG**	**LO**	**SR**	**IR**	**IP**	**MG**	**LO**	**SR**	**IR**	**IP**	**MG**	**LO**	**SR**
Barman	10	0	0	0	**0**	10	0	0	0	**0**	10	0	0	0	**0**
Blocksworkd	6	0	0	0	0.4	4	0	0	0	0.6	0	0	0	1	**1**
Floortile	9	0	1	0	**0**	9	1	0	0	**0**	3	7	0	0	**0**
Grippers	3	0	0	8	0.7	0	1	1	2	0.8	0	0	1	2	**0.9**
Tyreworld	9	0	0	N/A	0.1	8	0	0	N/A	0.2	0	0	0	N/A	**1**
Termes	10	0	0	N/A	**0**	10	0	0	N/A	**0**	10	0	0	N/A	**0**

Table 1: The error type count (IR, IP, MG, LO), as defined in Section 2. Note that LO only counts if the model outputs a feasible plan yet is not optimal. Besides, SR is the success rate, or feasible plan rate. The highest success rate for each domain is in bold. o1-preview has the highest success rate.

3 Discussion

3.1 Empirical Limitations

The primary limitation of this study stems from the relatively small dataset used in our empirical evaluations. While the experiments provide a foundational understanding of the o1 model's planning capabilities, broader insights into its generalizability and robustness can only be derived with more extensive testing across larger and more diverse datasets. Larger datasets would help expose potential weaknesses that remain hidden in smaller, more structured environments, and would allow us to explore how o1 models handle a wider variety of constraints and complexity levels. Future work shall evaluate on more data and in more real-world settings involving dynamic and less predictable elements.

3.2 Model Performance vs. Problem Complexity

Our analysis reveals a strong correlation between the complexity of the problem and the performance of the o1 model. We empirically examine each problem along two dimensions of complexity: action complexity and spatial complexity, as illustrated in Figure 2.

Specifically, the *Floortile* and *Termes* tasks highlight the challenges o1 faces in environments with higher spatial and rule-based complexity. In *Floortile*, the task is set in a two-dimensional world, where robots must follow strict painting rules while navigating a constrained grid. In contrast, *Termes* involves a three-dimensional setting, introducing additional layers of complexity due to vertical movement constraints and the need for precise block manipulation. Interestingly, the size of the action space did not appear to significantly affect the model's ability to capture and use context efficiently. Instead, the complexity of spatial relationships and state transitions proved to be more problematic. This suggests that while o1 models can handle tasks with limited actions (e.g., *Grippers*), they struggle when required to reason about more abstract, multi-dimensional spaces where maintaining an accurate internal state becomes critical.

3.3 Constraint Following and State Management

One key finding of this study is the o1 model's improved ability to follow constraints and manage states, especially in comparison to GPT-4. o1-preview's self-evaluation mechanism, which allows the model to check and correct its actions during plan generation, was particularly effective in tasks like *Blocksworld* and *Tyreworld*. In these tasks, o1-preview demonstrated a higher success rate in adhering to complex rules, such as the preconditions for using a wrench or jack in *Tyreworld*, while avoiding the rule violations that plagued GPT-4 and o1-mini. However, this ability to follow constraints deteriorates in more complex environments like *Termes*, where the need for precise spatial reasoning and multi-step manipulation often leads to rule violations and misinterpretations of task goals. This points to a potential limitation in the model's state management when dealing with more abstract problem spaces, and may call for explicit integration of neurosymbolic methods Yang et al. [2024a].

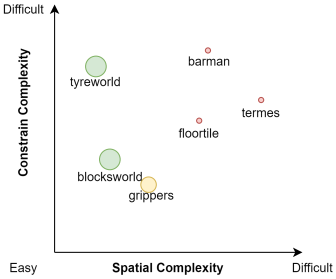


Figure 2: Empirical evaluation was conducted based on action complexity and spatial complexity. The color coding represents the success rates of the o1-preview model: green indicates a high success rate, yellow moderate success, and red complete failure.

3.4 Optimality and Redundancy in Planning

Optimality remains a significant challenge for the o1 models, as demonstrated across tasks like *Blocksworld* and *Floortile*. While o1-preview often generated feasible plans, it frequently failed to produce optimal solutions, leading to redundant actions and inefficiencies. For example, in *Blocksworld*, o1-preview added unnecessary steps to the plan, reducing its overall efficiency despite reaching the correct goal state. This suggests that while the model can understand and follow constraints, it struggles with decision-making related to resource minimization and action optimization. The ability to reason about optimality is crucial for real-world applications, where minimizing steps and resources is often as important as achieving the correct outcome. Enhancing this aspect of o1's reasoning mechanism—perhaps by incorporating more advanced cost-based decision frameworks—would be a valuable area for future research. Additionally, we observed that all three models exhibited some hallucination, including the assumption of non-existent rules. For instance, the o1-preview model in the grippers assumed that it could only move to adjacent numbered rooms, whereas the actual rule specifies that the move action can proceed to any room. Although model could still generate feasible plan, but that also hinder its ability to generate optimal plan.

3.5 Generalization and Adaptability

Another promising outcome of this study is o1-preview's demonstrated ability to generalize across tasks with consistent rule structures, as seen in *Grippers*. In these cases, o1-preview outperformed GPT-4 by effectively adapting its learned strategies to new environments. In these scenarios, o1-preview consistently outperformed GPT-4 by effectively adapting its learned strategies to new environments. The o1 model attempted to imbue meaningless symbols with natural language meaning to aid problem-solving, as seen in Figure 12. Additionally, o1-preview's self-evaluation capabilities enabled it to maintain reasonable adherence to constraints, with only minor deviations, compared to GPT-4, which often fails to grasp the goal. While o1-preview's generalizability surpasses GPT4, particularly in structured, low-dimensional tasks, there is still substantial room for improvement in enabling these models to adapt to more dynamic, high-dimensional, and abstract problem spaces.

4 Conclusion

We present the detailed analysis about our findings and future directions in Section B. In summary, while o1-preview represents a notable advancement in LLM-based planning, significant challenges remain, particularly in terms of optimizing plans, generalizing to more abstract tasks, and managing state complexity. Future research should aim to build on these insights to create more robust, efficient, and adaptable planning agents capable of handling the diverse range of challenges presented by real-world planning problems.

References

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022a.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022b.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*, 2024a.

Karthik Valmeekam, Sarath Sreedharan, Matthew Marquez, Alberto Olmo, and Subbarao Kambhampati. On the planning abilities of large language models (a critical investigation with a proposed benchmark). *arXiv preprint arXiv:2302.06706*, 2023.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.

OpenAI. Learning to reason with llms. `https://openai.com/index/learning-to-reason-with-llms/`, 2024.

Yunhao Yang, Neel P Bhatt, Tyler Ingebrand, William Ward, Steven Carr, Zhangyang Wang, and Ufuk Topcu. Fine-tuning language models using formal methods feedback: A use case in autonomous systems. *Proceedings of Machine Learning and Systems*, 6:339–350, 2024a.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, et al. Memory^3: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*, 2024b.

Wenqing Zheng, SP Sharan, Zhiwen Fan, Kevin Wang, Yihan Xi, and Zhangyang Wang. Symbolic visual reinforcement learning: A scalable framework with object-level abstraction and differentiable expression search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Lisong C Sun, Neel P Bhatt, Jonathan C Liu, Zhiwen Fan, Zhangyang Wang, Todd E Humphreys, and Ufuk Topcu. Mm3dgs slam: Multi-modal 3d gaussian splatting for slam using vision, depth, and inertial measurements. *arXiv preprint arXiv:2404.00923*, 2024.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

A Experiments on Planning Benchmarks

We assess the planning capabilities of GPT-4, o1-mini, and o1-preview as discussed in Section 2. Our code and experimental results are available at https://github.com/VITA-Group/o1-planning, with all models (GPT-4, o1-mini, and o1-preview) evaluated using their default settings. Here we manually evaluated the planning results to identify success or error patterns and conducted fine-grained analysis on each instance. We leave scalable automatic evaluation of planning for future work. The overall comparison in outlined in Table 1. Next, we will explore each task in detail to carefully analyze the language agents' abilities across various planning tasks. The results afor feasibility, optimality, and generalizability are presented in figures 3, 4, and 5, respectively. For each task, errors are highlighted in red, while the reasons for these errors are indicated in orange. The specific action constraints that are violated are discussed in the accompanying captions. Details about the action domain and additional examples for each problem will be released after acceptance.

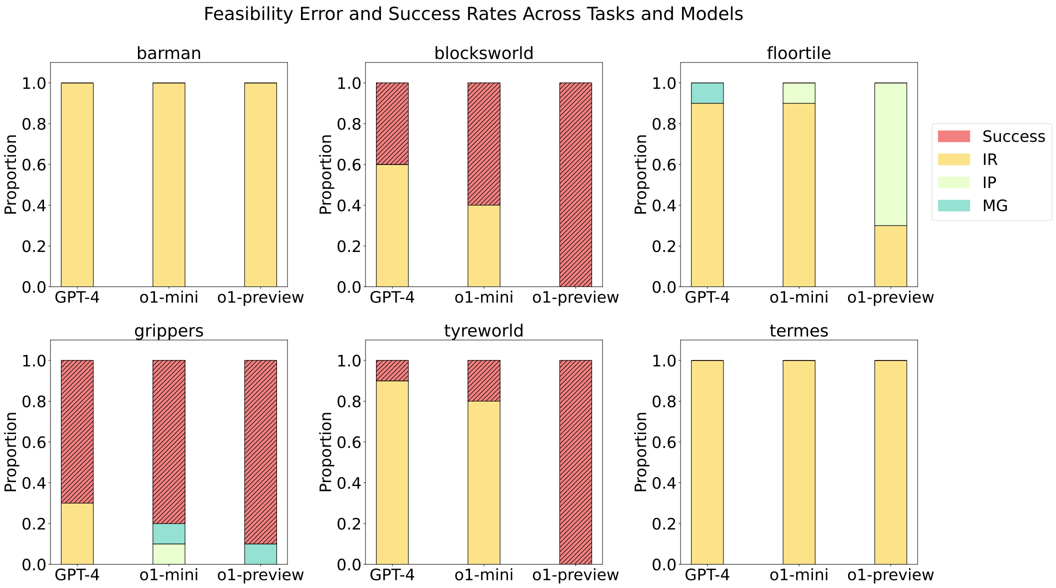


Figure 3: Feasibility error and success rate for 6 tasks and 3 models. Overall, o1 improves the success rate for certain tasks, but many problematic issues still persist. Examples of different error types are detailedd in later figures: IR: 6, 7a, 9, 10, 11 ; IP: 9 ; MG: 8a

.

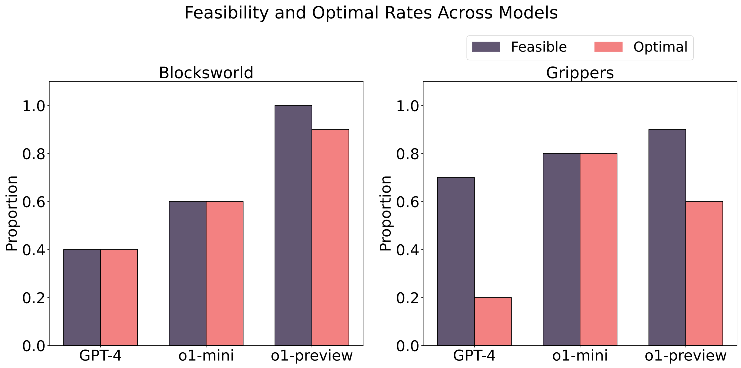


Figure 4: Success rate and optimality rate for *Blocksworld* and *Grippers*. Compared to GPT-4, o1 can provide more optimal plans. Example of suboptimal solutions are provided in Figures 7b and 8b.

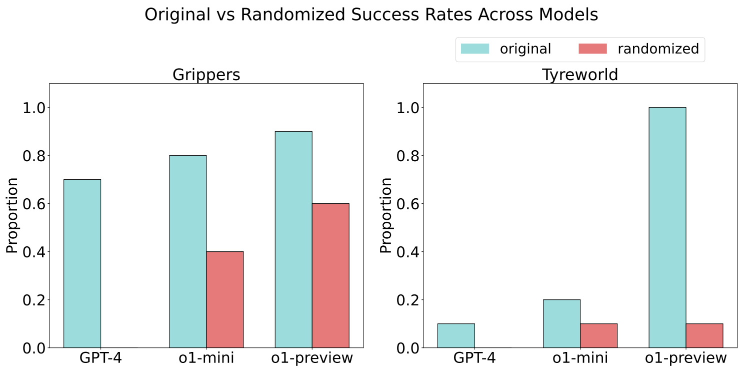
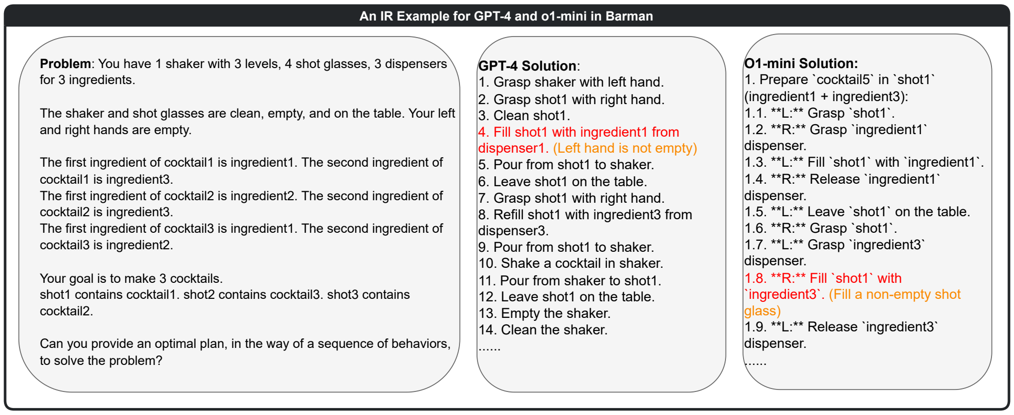


Figure 5: Success rate for generalization setting. GPT-4 fails entirely on challenging generalized tasks, whereas o1 is able to solve some of them. An example of randomize domain is provided in Figure 12.

A.1 Barman

Task description In this task, a robot barman is tasked with preparing a series of drinks by manipulating drink dispensers, shot glasses, and a shaker. The robot, equipped with two hands, must perform a variety of actions such as grasping containers, filling/refilling shot glasses, pouring ingredients, shaking cocktails, and cleaning or emptying containers. Each action comes with strict preconditions—for example, the robot can only grasp a container when one hand is free, and shaking a cocktail is only possible when the shaker contains exactly two ingredients. Successfully completing this task requires precise sequencing of actions, where adhering to these constraints is crucial to avoid mistakes.

Analysis The results indicate that the language model agent struggles significantly with this task, consistently failing to generate feasible plans. Nearly all the errors stem from the agent's inability to follow the specified rules, categorized as the IR error. For example, the rules demand that certain actions, like filling a container, require one hand to be free, or that specific actions must be taken in a strict order, such as holding a container before filling it. However, the LLM agents often overlook these critical constraints. This issue persists even in more advanced models like o1-preview. Figure 6 illustrates the first error in the solution, where both GPT-4 and o1-mini violate explicit rules, highlighted in red and orange, respectively. These errors underline a common limitation of language models: while they can generate plausible sequences of actions in a natural language context, they frequently overlook key operational constraints critical for real-world execution.



An IR Example for GPT-4 and o1-mini in Barman

Problem: You have 1 shaker with 3 levels, 4 shot glasses, 3 dispensers for 3 ingredients.

The shaker and shot glasses are clean, empty, and on the table. Your left and right hands are empty.

The first ingredient of cocktail1 is ingredient1. The second ingredient of cocktail1 is ingredient3.
The first ingredient of cocktail2 is ingredient2. The second ingredient of cocktail2 is ingredient3.
The first ingredient of cocktail3 is ingredient1. The second ingredient of cocktail3 is ingredient2.

Your goal is to make 3 cocktails.
shot1 contains cocktail1. shot2 contains cocktail3. shot3 contains cocktail2.

Can you provide an optimal plan, in the way of a sequence of behaviors, to solve the problem?

GPT-4 Solution:

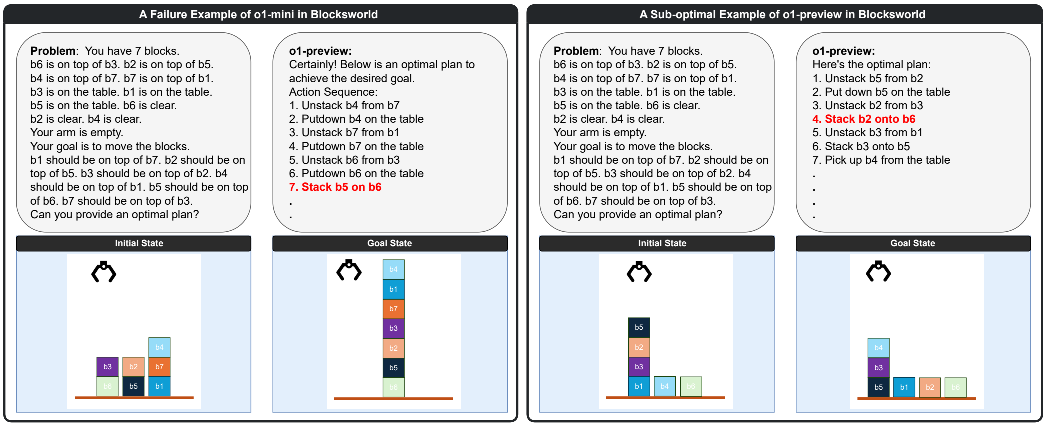
```
1. Grasp shaker with left hand.
2. Grasp shot1 with right hand.
3. Clean shot1.
4. Fill shot1 with ingredient1 from dispenser1. (Left hand is not empty)
5. Pour from shot1 to shaker.
6. Leave shot1 on the table.
7. Grasp shot1 with right hand.
8. Refill shot1 with ingredient3 from dispenser3.
9. Pour from shot1 to shaker.
10. Shake a cocktail in shaker.
11. Pour from shaker to shot1.
12. Leave shot1 on the table.
13. Empty the shaker.
14. Clean the shaker.
......
```

O1-mini Solution:

```
1. Prepare `cocktail5` in `shot1` (ingredient1 + ingredient3):
1.1. **L:** Grasp `shot1`.
1.2. **R:** Grasp `ingredient1` dispenser.
1.3. **L:** Fill `shot1` with `ingredient1`.
1.4. **R:** Release `ingredient1` dispenser.
1.5. **L:** Leave `shot1` on the table.
1.6. **R:** Grasp `shot1`.
1.7. **L:** Grasp `ingredient3` dispenser.
1.8. **R:** Fill `shot1` with `ingredient3`. (Fill a non-empty shot glass)
1.9. **L:** Release `ingredient3` dispenser.
......
```

Figure 6: A failure example for Barman. The left side contains the problem statement, while the right side shows the first lines of the solutions provided by GPT-4 and o1-mini. The GPT-4 solution fails because the rules require that one hand must be empty for "filling", while the o1-mini solution fails because the rules specify that "filling" applies only to an empty shot glass.

A.2 Blocksworld



A Failure Example of o1-mini in Blocksworld

Problem: You have 7 blocks.
b6 is on top of b3. b2 is on top of b5.
b4 is on top of b7. b7 is on top of b1.
b3 is on the table. b1 is on the table.
b5 is on the table. b6 is clear.
b2 is clear. b4 is clear.
Your arm is empty.
Your goal is to move the blocks.
b1 should be on top of b7. b2 should be on top of b5. b3 should be on top of b2. b4 should be on top of b1. b5 should be on top of b6. b7 should be on top of b3.
Can you provide an optimal plan?

o1-preview:
Certainly! Below is an optimal plan to achieve the desired goal.
Action Sequence:
1. Unstack b4 from b7
2. Putdown b4 on the table
3. Unstack b7 from b1
4. Putdown b7 on the table
5. Unstack b6 from b3
6. Putdown b6 on the table
7. Stack b5 on b6
.
.

A Sub-optimal Example of o1-preview in Blocksworld

Problem: You have 7 blocks.
b6 is on top of b3. b2 is on top of b5.
b4 is on top of b7. b7 is on top of b1.
b3 is on the table. b1 is on the table.
b5 is on the table. b6 is clear.
b2 is clear. b4 is clear.
Your arm is empty.
Your goal is to move the blocks.
b1 should be on top of b7. b2 should be on top of b5. b3 should be on top of b2. b4 should be on top of b1. b5 should be on top of b6. b7 should be on top of b3.
Can you provide an optimal plan?

o1-preview:
Here's the optimal plan:
1. Unstack b5 from b2
2. Put down b5 on the table
3. Unstack b2 from b3
4. Stack b2 onto b6
5. Unstack b3 from b1
6. Stack b3 onto b5
7. Pick up b4 from the table
.
.
.
.

(a)

(b)

Figure 7: Example failure and sub-optimal cases in the Blocksworld planning problem. (a) An illustration of a plan generated by o1-mini showcasing IR error in step 7. (b) An illustration of a sub-optimal plan generated by o1-preview showcasing LO error in step 4.

Task description This planning task involves multiple blocks arranged on a table, where the goal is to move from an initial configuration to a pre-specified goal configuration. The robot arm, which can hold only one block at a time, must execute a series of actions such as picking up, putting down, stacking, and unstacking blocks to achieve the desired arrangement. The challenge lies in determining the correct sequence of these actions, while adhering to constraints that dictate how blocks can be manipulated. The action space is limited to fundamental operations like pickup, putdown, and stack.

Analysis In this task, GPT-4 demonstrated a relatively low success rate of 40%, while o1-mini performed slightly better at 60%. However, o1-preview achieved a perfect 100% success rate, reflecting its stronger reasoning capabilities. The success rates decreased as the number of blocks increased, highlighting the challenge of managing larger object sets. Both GPT-4 and o1-mini frequently failed to follow problem constraints, an issue categorized as the IR error. Figure 7a provides an example where o1-mini fails to comply with these constraints.

Although o1-preview successfully completed all tasks, it was not entirely optimal. In one instance, the model added an unnecessary step, leading to a suboptimal solution despite reaching the correct goal state. This issue, characterized as LO, is illustrated in Figure 7b. The occurrence of suboptimal steps even in successful models emphasizes the ongoing challenge of optimizing planning tasks, where generating feasible solutions is not always sufficient.

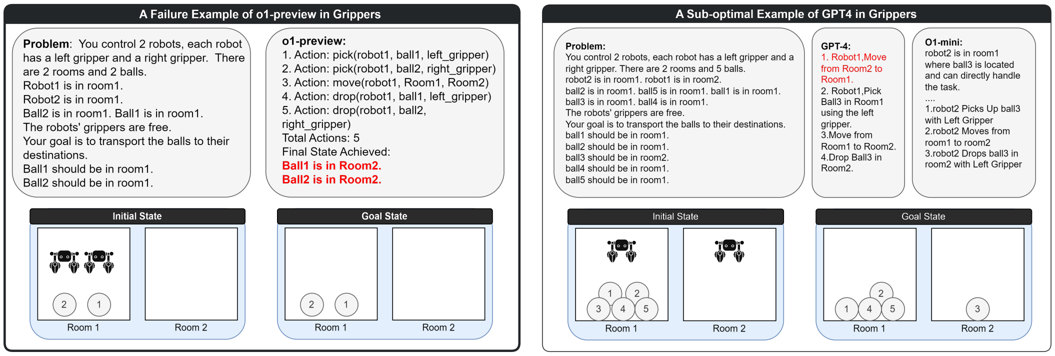
A.3 Grippers

Task description This task involves a team of robots equipped with two grippers, capable of moving between rooms and manipulating objects. The robots have three primary actions: moving from one room to another, picking up objects, and dropping them. Each action is constrained by the robot's current location and the status of its grippers, meaning that a robot can only pick up an object if its gripper is free and can only drop an object in a specific location once it is carrying one. Effective planning requires coordinating these actions while adhering to these constraints to accomplish the goal of manipulating objects across different rooms.

Analysis In this domain, both o1-mini and o1-preview significantly outperformed GPT-4, particularly in success and optimality rates. GPT-4 managed a 70% success rate but only a 20% optimality rate, indicating frequent suboptimal action sequences. In contrast, o1-mini achieved both higher success and optimality rates, at 80% for each. o1-preview performed even better with a 90% success rate, though its optimality rate dropped slightly to 70%. Figure 8b showcases an example where

GPT-4 generates a suboptimal solution by including an unnecessary relocation of the robot, which adds redundant steps to the plan. Conversely, o1-mini, through additional reasoning, eliminated unnecessary movements, leading to an optimal solution.

However, o1-preview exhibited a shortcoming related to the MG error. In one instance, depicted in Figure 8a, the robot's initial state already coincided with the goal state—both balls were already in the correct room (room1), meaning no action was required. While both GPT-4 and o1-mini correctly recognized that no further steps were necessary, o1-preview mistakenly identified room2 as the goal, leading to unnecessary actions aimed at relocating the balls. This misinterpretation demonstrates that even advanced models like o1-preview can struggle with maintaining accurate goal-state representations, especially in tasks requiring minimal or no action.



A Failure Example of o1-preview in Grippers

Problem: You control 2 robots, each robot has a left gripper and a right gripper. There are 2 rooms and 2 balls.
Robot1 is in room1.
Robot2 is in room1.
Ball2 is in room1. Ball1 is in room1.
The robots' grippers are free.
Your goal is to transport the balls to their destinations.
Ball1 should be in room1.
Ball2 should be in room1.

o1-preview:
1. Action: pick(robot1, ball1, left_gripper)
2. Action: pick(robot1, ball2, right_gripper)
3. Action: move(robot1, Room1, Room2)
4. Action: drop(robot1, ball1, left_gripper)
5. Action: drop(robot1, ball2, right_gripper)
Total Actions: 5
Final State Achieved:
Ball1 is in Room2.
Ball2 is in Room2.

A Sub-optimal Example of GPT4 in Grippers

Problem:
You control 2 robots, each robot has a left gripper and a right gripper. There are 2 rooms and 5 balls.
robot2 is in room1. robot1 is in room2.
ball2 is in room1. ball5 is in room1. ball1 is in room1.
ball3 is in room1. ball4 is in room1.
The robots' grippers are free.
Your goal is to transport the balls to their destinations.
ball1 should be in room1.
ball2 should be in room1.
ball3 should be in room2.
ball4 should be in room1.
ball5 should be in room1.

GPT-4:
1. Robot1,Move from Room2 to Room1.
2. Robot1,Pick Ball3 in Room1 using the left gripper.
3.Move from Room1 to Room2.
4.Drop Ball3 in Room2.

O1-mini:
robot2 is in room1 where ball3 is located and can directly handle the task.
....
1.robot2 Picks Up ball3 with Left Gripper
2.robot2 Moves from room1 to room2
3.robot2 Drops ball3 in room2 with Left Gripper

(a)

(b)

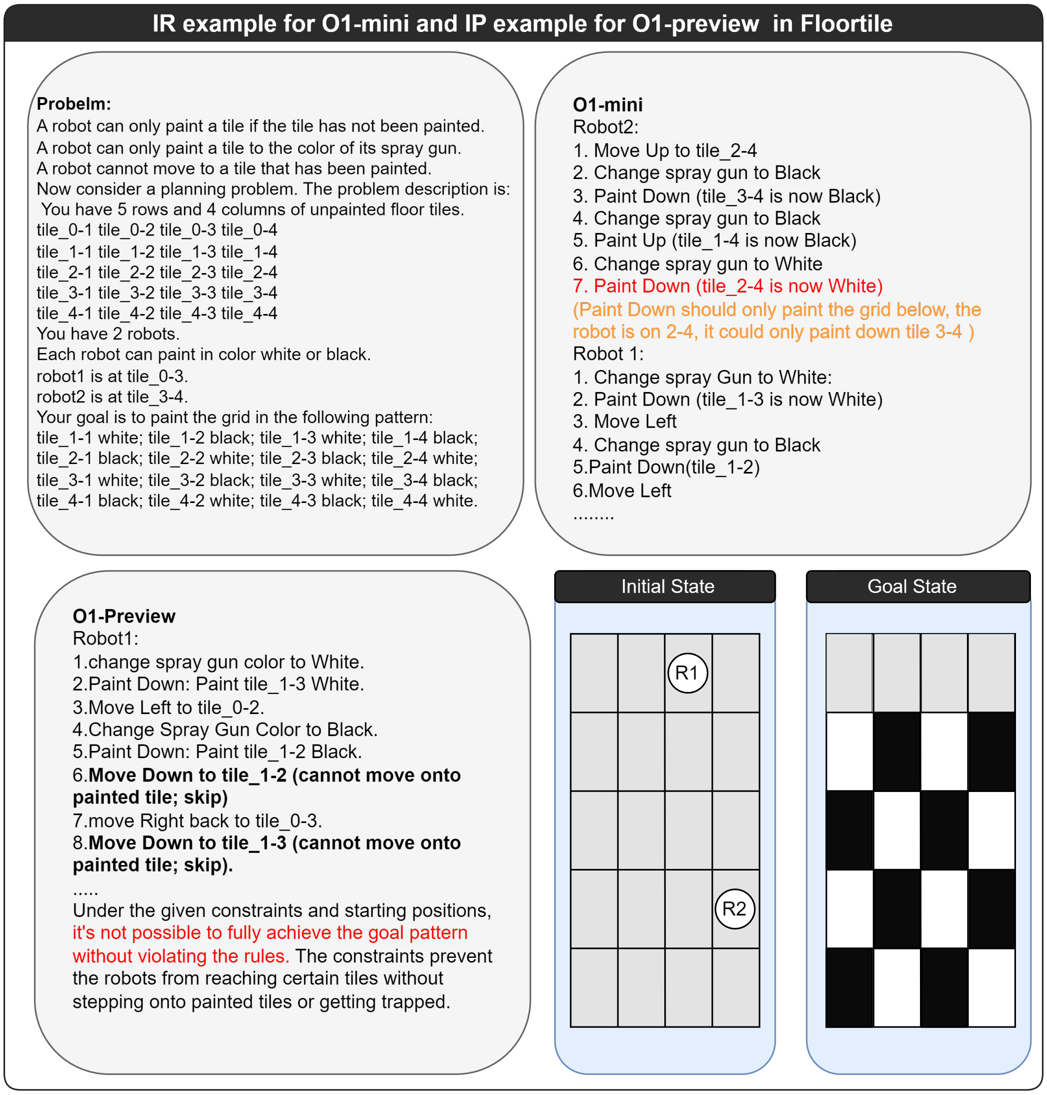
Figure 8: Failure examples in Grippers. (a) showcases o1-preview MG error: o1-preview assumes the goal state is both balls in room2 instead of room1; (b) highlights GPT4's suboptimality: it took an extra step to complete the goal, while o1-mini can return the optimal plan.

A.4 Floortile

Task description In this task, a team of robots is responsible for painting a grid of floor tiles in black and white. Each robot can move in four directions, switch the color of its spray gun, and paint tiles directly in front of or behind them. The main challenge is that robots can only paint tiles that are currently unpainted and cannot move onto tiles that have already been painted. This creates a complex constraint, requiring careful planning of movements and actions to achieve the desired tile pattern without the robots trapping themselves or each other. The task demands strategic coordination between movement and painting actions, ensuring the robots follow the rules while efficiently completing the grid pattern.

Analysis In this domain, all models—GPT-4, o1-mini, and o1-preview—failed to solve the test cases, but the reasons for their failures varied. For GPT-4 and o1-mini, 90% of their failures stemmed from the IR error. Specifically, both models frequently violated the rule that robots can only paint tiles directly in front or behind them, instead attempting to paint the tile on which they were standing. This rule violation was a common source of errors as the models struggled to keep track of the task constraints while moving and painting simultaneously.

On the other hand, o1-preview showed a notable improvement in this regard, with only 30% of its failures caused by IR. o1-preview's internal self-evaluation mechanism allowed it to better track the rules and adjust its actions accordingly. For instance, when it initially attempted to paint the wrong tile, it was able to reevaluate the action and correct itself by following the task constraints. However, despite these improvements in rule adherence, o1-preview encountered other errors, such as rule confusion. In some cases, it misinterpreted which tiles could be painted or made invalid assumptions about the sequence of movements. While its chain-of-thought reasoning helped it self-correct in some cases, it was ultimately unable to solve the task entirely, as seen in Figure 9.



IR example for O1-mini and IP example for O1-preview in Floortile

Probelm:
A robot can only paint a tile if the tile has not been painted.
A robot can only paint a tile to the color of its spray gun.
A robot cannot move to a tile that has been painted.
Now consider a planning problem. The problem description is:
You have 5 rows and 4 columns of unpainted floor tiles.
tile_0-1 tile_0-2 tile_0-3 tile_0-4
tile_1-1 tile_1-2 tile_1-3 tile_1-4
tile_2-1 tile_2-2 tile_2-3 tile_2-4
tile_3-1 tile_3-2 tile_3-3 tile_3-4
tile_4-1 tile_4-2 tile_4-3 tile_4-4
You have 2 robots.
Each robot can paint in color white or black.
robot1 is at tile_0-3.
robot2 is at tile_3-4.
Your goal is to paint the grid in the following pattern:
tile_1-1 white; tile_1-2 black; tile_1-3 white; tile_1-4 black;
tile_2-1 black; tile_2-2 white; tile_2-3 black; tile_2-4 white;
tile_3-1 white; tile_3-2 black; tile_3-3 white; tile_3-4 black;
tile_4-1 black; tile_4-2 white; tile_4-3 black; tile_4-4 white.

O1-mini
Robot2:
1. Move Up to tile_2-4
2. Change spray gun to Black
3. Paint Down (tile_3-4 is now Black)
4. Change spray gun to Black
5. Paint Up (tile_1-4 is now Black)
6. Change spray gun to White
7. Paint Down (tile_2-4 is now White)
(Paint Down should only paint the grid below, the robot is on 2-4, it could only paint down tile 3-4)
Robot 1:
1. Change spray Gun to White:
2. Paint Down (tile_1-3 is now White)
3. Move Left
4. Change spray gun to Black
5.Paint Down(tile_1-2)
6.Move Left
....

O1-Preview
Robot1:
1.change spray gun color to White.
2.Paint Down: Paint tile_1-3 White.
3.Move Left to tile_0-2.
4.Change Spray Gun Color to Black.
5.Paint Down: Paint tile_1-2 Black.
6.**Move Down to tile_1-2 (cannot move onto painted tile; skip)**
7.move Right back to tile_0-3.
8.**Move Down to tile_1-3 (cannot move onto painted tile; skip).**
.....
Under the given constraints and starting positions, it's not possible to fully achieve the goal pattern without violating the rules. The constraints prevent the robots from reaching certain tiles without stepping onto painted tiles or getting trapped.

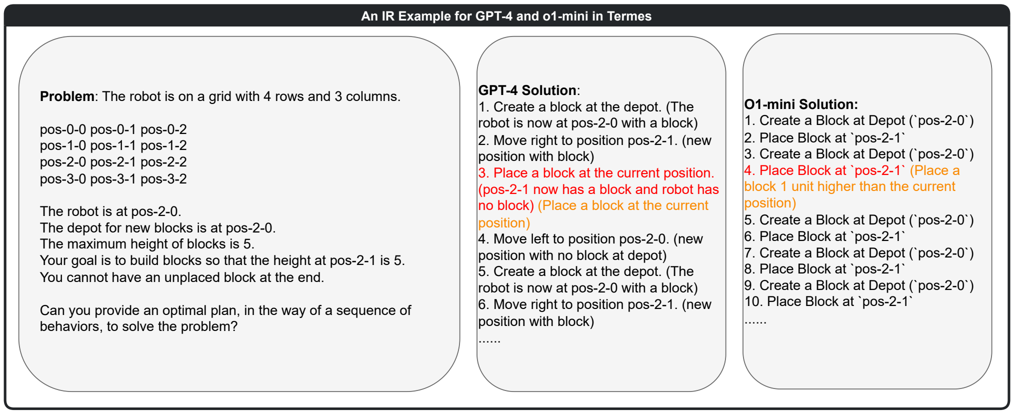
Figure 9: We illustrate the IR error of o1-mini (top right) and the IP error of o1-preview (bottom left). o1-mini incorrectly assumed that "paint down" would paint the grid beneath the robot. Meanwhile, o1-preview adhered to the constraints throughout the plan but ultimately failed to complete it.

A.5 Termes

Task description The Termes task requires controlling a robot to construct structures by moving between different positions and manipulating blocks. The robot can move horizontally, vertically (up and down), and is tasked with placing or removing blocks at neighboring positions that match in height. Additionally, the robot can create new blocks at a depot or destroy blocks when needed. To achieve the specified construction goals, the robot must efficiently plan its movements and strategically use blocks, adhering to height and positional constraints throughout the task. The challenge lies in coordinating these actions while respecting the rules that govern both movement and block placement.

Analysis All models—GPT-4, o1-mini, and o1-preview—failed to successfully complete the tasks in the Termes domain, largely due to shortcomings in detailed planning. A common source of error across all models was the failure to account for the task's height constraints when moving horizontally, upward, or downward. These constraints, which ensure that the robot can only move to positions of matching or specific relative heights, were frequently ignored by the models. Additionally, the language models often made the mistake of placing blocks at their current position, a violation of the task's rule that blocks can only be placed at neighboring positions.

These errors highlight the models' difficulty in managing complex spatial relationships and adhering to the intricate rules of the task. Although the actions generated by the models may seem plausible in a natural language context, they frequently overlook key operational details required for real-world robot planning. This inability to follow task-specific constraints leads to failures in execution, as illustrated in Figure 10, where the model skips necessary steps and misinterprets the correct sequence of actions.



An IR Example for GPT-4 and o1-mini in Termes

Problem: The robot is on a grid with 4 rows and 3 columns.

pos-0-0 pos-0-1 pos-0-2
pos-1-0 pos-1-1 pos-1-2
pos-2-0 pos-2-1 pos-2-2
pos-3-0 pos-3-1 pos-3-2

The robot is at pos-2-0.
The depot for new blocks is at pos-2-0.
The maximum height of blocks is 5.
Your goal is to build blocks so that the height at pos-2-1 is 5.
You cannot have an unplaced block at the end.

Can you provide an optimal plan, in the way of a sequence of behaviors, to solve the problem?

GPT-4 Solution:
1. Create a block at the depot. (The robot is now at pos-2-0 with a block)
2. Move right to position pos-2-1. (new position with block)
3. Place a block at the current position. (pos-2-1 now has a block and robot has no block) (Place a block at the current position)
4. Move left to position pos-2-0. (new position with no block at depot)
5. Create a block at the depot. (The robot is now at pos-2-0 with a block)
6. Move right to position pos-2-1. (new position with block)
......

O1-mini Solution:
1. Create a Block at Depot (`pos-2-0`)
2. Place Block at `pos-2-1`
3. Create a Block at Depot (`pos-2-0`)
4. Place Block at `pos-2-1` (Place a block 1 unit higher than the current position)
5. Create a Block at Depot (`pos-2-0`)
6. Place Block at `pos-2-1`
7. Create a Block at Depot (`pos-2-0`)
8. Place Block at `pos-2-1`
9. Create a Block at Depot (`pos-2-0`)
10. Place Block at `pos-2-1`
......

Figure 10: A failure example for Termes. GPT-4 solution fails because the block can only be placed in a neighboring position, while o1-mini solution fails since the block must be placed at the same height.

A.6 Tyreworld

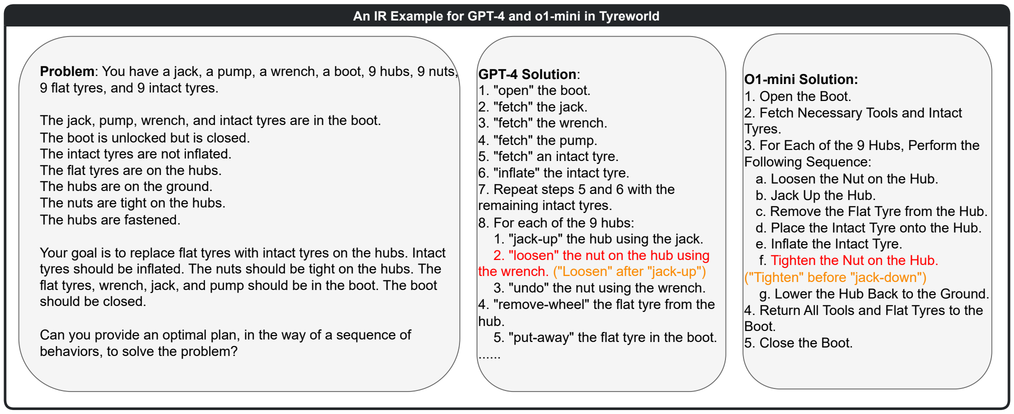
Task description This task involves replacing flat tyres on vehicle hubs with intact, inflated tyres. The process requires the use of tools such as a wrench, jack, and pump, and the agent must follow specific actions to manipulate the tyres, nuts, and tools. There are 11 predefined actions, including opening and closing the boot, fetching and storing tools, loosening and tightening nuts, jacking up and down hubs, removing and installing wheels, inflating tyres, and securing or undoing nuts. Success in this task depends on executing these actions in the correct sequence while satisfying specific preconditions, such as using the wrench to loosen the nuts before removing the wheel or tightening the nuts only after lowering the jack.

Analysis o1-preview generated correct plans for all test problems, significantly outperforming both GPT-4 and o1-mini, which failed to complete all but the simplest cases. The primary issue for GPT-4 and o1-mini was their frequent failure to follow the required action sequences. For instance, common errors included "loosening the nuts after jacking up" or "tightening the nuts before jacking down"—critical mistakes that would prevent successful tyre replacement. Figure 11 provides an example where failing to follow these mechanical constraints leads to incorrect plans.

We also evaluated the models' generalization capabilities within this domain, revealing that while o1-preview performed well on structured tasks, its success rate dropped from 100% to 20% when the actions and tools were replaced with random symbols. Despite the symbols being abstract, the underlying action constraints and logic remained unchanged. This sharp decrease in performance, as illustrated in Figure 12, suggests that while o1-preview excels in rule-based planning with familiar symbols, it struggles to generalize when the problem context becomes more abstract.

B Findings and Future Directions

Our study offers a pilot evaluation of the planning capabilities of OpenAI's o1 models, providing new insights into their strengths and limitations. By systematically evaluating their feasibility, optimality, and generalizability across diverse planning tasks, we have uncovered key areas where



An IR Example for GPT-4 and o1-mini in Tyreworld

Problem: You have a jack, a pump, a wrench, a boot, 9 hubs, 9 nuts, 9 flat tyres, and 9 intact tyres.

The jack, pump, wrench, and intact tyres are in the boot.
The boot is unlocked but is closed.
The intact tyres are not inflated.
The flat tyres are on the hubs.
The hubs are on the ground.
The nuts are tight on the hubs.
The hubs are fastened.

Your goal is to replace flat tyres with intact tyres on the hubs. Intact tyres should be inflated. The nuts should be tight on the hubs. The flat tyres, wrench, jack, and pump should be in the boot. The boot should be closed.

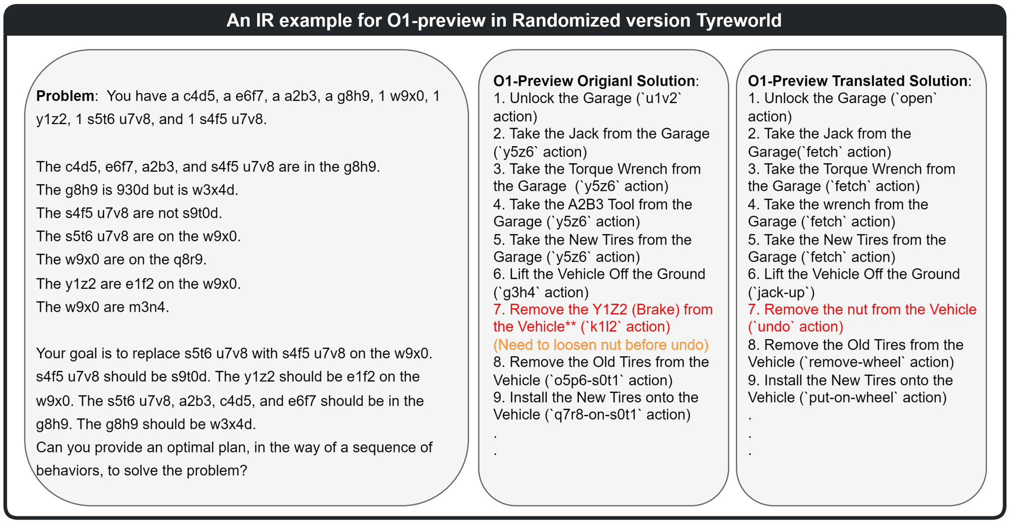
Can you provide an optimal plan, in the way of a sequence of behaviors, to solve the problem?

GPT-4 Solution:
1. "open" the boot.
2. "fetch" the jack.
3. "fetch" the wrench.
4. "fetch" the pump.
5. "fetch" an intact tyre.
6. "inflate" the intact tyre.
7. Repeat steps 5 and 6 with the remaining intact tyres.
8. For each of the 9 hubs:
   1. "jack-up" the hub using the jack.
   2. "loosen" the nut on the hub using the wrench. ("Loosen" after "jack-up")
   3. "undo" the nut using the wrench.
4. "remove-wheel" the flat tyre from the hub.
   5. "put-away" the flat tyre in the boot.

……

O1-mini Solution:
1. Open the Boot.
2. Fetch Necessary Tools and Intact Tyres.
3. For Each of the 9 Hubs, Perform the Following Sequence:
   a. Loosen the Nut on the Hub.
   b. Jack Up the Hub.
   c. Remove the Flat Tyre from the Hub.
   d. Place the Intact Tyre onto the Hub.
   e. Inflate the Intact Tyre.
   f. Tighten the Nut on the Hub. ("Tighten" before "jack-down")
   g. Lower the Hub Back to the Ground.
4. Return All Tools and Flat Tyres to the Boot.
5. Close the Boot.

Figure 11: A failure example for Tyreworld. GPT-4 fails since "loosen" must occur before "jack-up", and o1-mini fails since "tighten" must happen after "jack-down".



An IR example for O1-preview in Randomized version Tyreworld

Problem: You have a c4d5, a e6f7, a a2b3, a g8h9, 1 w9x0, 1 y1z2, 1 s5t6 u7v8, and 1 s4f5 u7v8.

The c4d5, e6f7, a2b3, and s4f5 u7v8 are in the g8h9.
The g8h9 is 930d but is w3x4d.
The s4f5 u7v8 are not s9t0d.
The s5t6 u7v8 are on the w9x0.
The w9x0 are on the q8r9.
The y1z2 are e1f2 on the w9x0.
The w9x0 are m3n4.

Your goal is to replace s5t6 u7v8 with s4f5 u7v8 on the w9x0. s4f5 u7v8 should be s9t0d. The y1z2 should be e1f2 on the w9x0. The s5t6 u7v8, a2b3, c4d5, and e6f7 should be in the g8h9. The g8h9 should be w3x4d.
Can you provide an optimal plan, in the way of a sequence of behaviors, to solve the problem?

O1-Preview Origianl Solution:
1. Unlock the Garage (`u1v2` action)
2. Take the Jack from the Garage (`y5z6` action)
3. Take the Torque Wrench from the Garage (`y5z6` action)
4. Take the A2B3 Tool from the Garage (`y5z6` action)
5. Take the New Tires from the Garage (`y5z6` action)
6. Lift the Vehicle Off the Ground (`g3h4` action)
7. Remove the Y1Z2 (Brake) from the Vehicle** (`k1l2` action) (Need to loosen nut before undo)
8. Remove the Old Tires from the Vehicle (`o5p6-s0t1` action)
9. Install the New Tires onto the Vehicle (`q7r8-on-s0t1` action)

.

.

O1-Preview Translated Solution:
1. Unlock the Garage (`open` action)
2. Take the Jack from the Garage(`fetch` action)
3. Take the Torque Wrench from the Garage (`fetch` action)
4. Take the wrench from the Garage (`fetch` action)
5. Take the New Tires from the Garage (`fetch` action)
6. Lift the Vehicle Off the Ground (`jack-up`)
7. Remove the nut from the Vehicle (`undo` action)
8. Remove the Old Tires from the Vehicle (`remove-wheel` action)
9. Install the New Tires onto the Vehicle (`put-on-wheel` action)

.

.

.

Figure 12: An example from the Randomized Tyre World: The center panel displays O-1's original solution, while the right panel translates the random symbols back to their original terms. Here, O-1 failed to adhere to the constraint that the nut must be loosened before it can be undone.

o1-preview demonstrates promising advancements as well as significant challenges that remain to be addressed.

B.1 Summary of Findings

The findings of our experiments can be summarized from four key perspectives:

1. **Understanding the Problem:** o1-preview demonstrated an improved ability to grasp the task requirements and constraints, particularly in well-defined, rule-based environments like *Barman* and *Tyreworld*. This was largely due to its self-evaluation mechanism, which allowed for more accurate state tracking and constraint adherence. However, more evidence is needed to establish whether these improvements translate to better reasoning capabilities in more abstract settings.
2. **Following Constraints:** Across most tasks, o1-preview showed a superior capacity to follow task-specific constraints compared to GPT-4. However, this ability weakened as the complexity of spatial reasoning and state transitions increased, as seen in *Termes*. This suggests that while constraint following is a relative strength of the o1 model, more work is needed to handle environments with higher-dimensional state spaces.

3. **State and Memory Management:** One of o1-preview's key advantages over previous models is its ability to remember and manage multiple states effectively within a plan, which contributed to its higher success rate in certain tasks. However, as problem complexity increased, the model's state management became less reliable, particularly in tasks involving spatial reasoning across multiple dimensions. This implies a potential bottleneck in the model's memory and decision-making processes.
4. **Reasoning and Generalization:** While o1-preview showed some promise in its generalization ability, particularly in structured environments like *Grippers*, its performance in more abstract tasks like *Termes* revealed substantial limitations. The model struggled with reasoning under conditions where actions and outcomes were less directly tied to the natural language representation of the task, highlighting an area for future improvements.

B.2 Opportunities for Improvement

We posit several key areas where future iterations of LLM-based planners can be improved:

- **Optimality and Resource Utilization:** Developing more sophisticated decision-making mechanisms that minimize redundant actions and optimize resource usage will be crucial for making o1 models more applicable to real-world planning tasks. This could involve incorporating cost-based reasoning or learning from expert demonstrations to achieve more optimal plans.

  Additionally, Retrieval-Augmented Generation (RAG) methods could offer a potential solution by providing real-time, low-cost external memory updates, especially when tasks rely on large knowledge bases encoded in natural language text. However, RAG's effectiveness hinges on the accuracy and efficiency of its retrieval algorithms, which may introduce further challenges.
- **Generalization in Abstract Spaces:** While o1-preview shows promise in generalizing across structured environments, its performance in tasks with more abstract and complex rule sets remains suboptimal. Future work should focus on enhancing the model's ability to generalize in high-dimensional and spatially dynamic environments, potentially through improved memory management Yang et al. [2024b] and abstraction mechanisms Zheng et al. [2024].

  Enhancing the model's decision-making and memory management capabilities, particularly for spatially complex tasks, will be essential for improving both optimality and generalizability in future iterations of LLM-based planning models.
- **Handling Dynamic and Unpredictable Environments:** Many real-world planning problems involve dynamic environments with unpredictable elements. Testing the o1 models in such settings would provide valuable insights into their robustness and adaptability, especially when rules or constraints change during execution.
- **Improving Constraint Adherence through Self-Evaluation:** One recurring issue across multiple domains is the models' inability to follow task-specific constraints accurately. Introducing more robust self-evaluation mechanisms could help LLMs better verify their own outputs before finalizing decisions, potentially catching mistakes like rule violations. Techniques such as multi-stage validation or symbolic verification Yang et al. [2024a], where models cross-check their proposed actions against the task constraints, could significantly reduce the incidence of constraint-related errors.
- **Leveraging Multimodal Inputs:** To enhance the model's understanding of spatial and physical reasoning tasks, future LLM-based planners could benefit from integrating multimodal inputs such as visual data, 3D environments, or sensor information Sun et al. [2024]. By incorporating non-textual data, planners would be better equipped to handle complex tasks, such as robotic manipulation or navigation, where purely text-based reasoning might miss critical spatial relationships or physical constraints.
- **Scalability to Complex Multi-Agent Planning:** Many planning tasks, particularly in robotics and logistics, require coordination between multiple agents Wu et al. [2023]. Extending LLM-based planners to effectively handle multi-agent systems would be an important step forward. This could involve developing strategies for decentralized planning,

where each agent generates its own plan based on local knowledge, while still cooperating to achieve a shared goal.

- **Incorporating Human Feedback for Continuous Learning:** One way to improve both optimality and generalization is by incorporating continuous learning through human feedback. Interactive feedback loops, where human users provide corrective signals or suggestions during plan execution, could help models refine their decision-making and better adapt to new situations or tasks that deviate from their training data.
- **Broader Impacts:** Our systematic analysis of LLM planning capabilities has implications for both research and real-world applications. The primary impact lies in understanding the limitations of LLM-based planning systems, particularly in spatial reasoning and state management, which is crucial for responsible deployment in critical applications like robotics and healthcare. While our evaluation itself poses minimal risks, over-reliance on LLM planners despite their identified limitations could lead to safety risks or operational failures in critical scenarios. We recommend implementing robust testing protocols before deploying these systems in high-stakes applications.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have clearly stated the contributions and scope in the Abstract and Introduction.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We describe the limitations in the Section 3.1.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have presented the information in Section A.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have the code and result released https://github.com/VITA-Group/o1-planning as stated in section A.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The evluation details are specified in Section A.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We released all the results and provided a detailed analysis of the results.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiments rely on the ChatGPT API service, and a standard computer should be capable of running them without issues.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform the NeurIPS Code of Ethics.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discusses the broader impact in Section B.2.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The released codes does not have a high risk for misuse.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite corresponding papers for the asserts we use in Section 1.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This paper utilizes existing problems for evaluation. We documented the changes we made to the assets.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.