Can LLM-Reasoning Models Replace Classical Planning? A Benchmark Study

A Preprint

Kai Goebel
Center for Vision, Automation & Control
AIT Austrian Institute of Technology GmbH
Vienna, Austria
kai.goebel@ait.ac.at

Patrik Zips
Center for Vision, Automation & Control
AIT Austrian Institute of Technology GmbH
Vienna, Austria
patrik.zips@ait.ac.at

August 1, 2025

Abstract

Recent advancements in Large Language Models have sparked interest in their potential for robotic task planning. While these models demonstrate strong generative capabilities, their effectiveness in producing structured and executable plans remains uncertain. This paper presents a systematic evaluation of a broad spectrum of current state of the art language models, each directly prompted using Planning Domain Definition Language domain and problem files, and compares their planning performance with the Fast Downward planner across a variety of benchmarks. In addition to measuring success rates, we assess how faithfully the generated plans translate into sequences of actions that can actually be executed, identifying both strengths and limitations of using these models in this setting. Our findings show that while the models perform well on simpler planning tasks, they continue to struggle with more complex scenarios that require precise resource management, consistent state tracking, and strict constraint compliance. These results underscore fundamental challenges in applying language models to robotic planning in real world environments. By outlining the gaps that emerge during execution, we aim to guide future research toward combined approaches that integrate language models with classical planners in order to enhance the reliability and scalability of planning in autonomous robotics.

Keywords Large Language Models · PDDL · Planning · Robotics · Hybrid Planning · Autonomous Agents

1 Introduction

Task planning has long been a central pillar in robotic systems, enabling autonomous agents to deliberate about sequences of actions, allocate resources, and adapt to changing constraints. Traditionally, symbolic planners such as Fast Downward Helmert [2006] have been the primary tools for generating provably correct solutions from well-defined domains using the Planning Domain Definition Language (PDDL) McDermott et al. [1998]. However, their reliance on precisely modeled PDDL files often leads to scalability and representation challenges when confronted with real-world uncertainty.

Meanwhile, Large Language Models (LLMs) such as GPT-4, Claude, and Llama variants have demonstrated promising capabilities in commonsense reasoning, which can be further enhanced through strategies like chain-of-thought prompting Wei et al. [2022]. These reasoning abilities enable LLMs to generate structured multi-step outputs, making them useful for high-level planning and even machine control. Recent studies show that LLMs can produce plans for simpler tasks, leveraging their extensive pretraining on textual corpora. For instance, approaches like ReAct Yao et al. [2023] and LLM-Planner Song et al. [2023] illustrate how LLMs can autonomously generate high-level action plans in open-domain contexts.

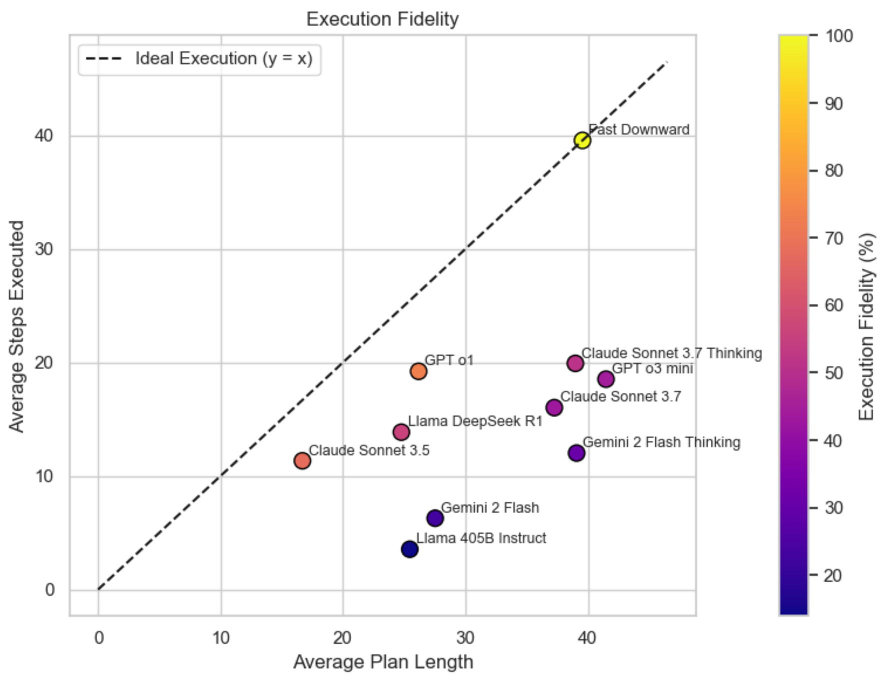

Figure 1: This scatter plot illustrates the relationship between average plan length (x-axis) and average executed steps (y-axis) for different LLM-based planners. The dashed diagonal ($y = x$) represents perfect execution fidelity, where all planned actions are successfully performed. Planners closer to this line exhibit better execution reliability. *Fast Downward* aligns with the ideal execution line, indicating that all planned actions are executable. *GPT-o1* has the highest execution fidelity (73.4%), followed by *Claude Sonnet 3.5* (67.9%), both performing well in executing planned actions. In contrast, *Llama DeepSeek R1* (55.9%) and *Claude Sonnet 3.7 Thinking* (51.1%) demonstrate moderate execution fidelity, while *Gemini 2 Flash* (22.8%) and *Llama 405B Instruct* (13.9%) exhibit significant execution failures, indicating poor planning quality. One can also see that *GPT-o3 mini* is the model that tends to overestimate the required plan lengths.

Despite these advancements, reliability and executability remain pressing concerns. LLMs often lack robust environment grounding Gramopadhye and Szafir [2023], Ahn et al. [2022] and are prone to generating incorrect or misleading outputs. This issue is especially critical for robotics, where a single erroneous action can risk safety or cause task failure. Moreover, there is growing recognition that domain constraints, partial observability, and concurrency require more than just fluent text generation. As Kambhampati Kambhampati et al. [2024], Kambhampati [2024] points out, pure LLM-based outputs require external validation or symbolic checks to ensure correctness in non-trivial planning scenarios.

These converging trends, powerful but imperfect Large Language Model planners on one hand and proven classical planners on the other, have led to the development of hybrid solutions. Several studies propose bridging the generative strengths of Large Language Models, particularly for domain knowledge and heuristics, with the search rigor of traditional planning tools Mahdavi et al. [2024] Liu et al. [2023], often yielding better results in complex or partially observable tasks. This synergy leverages the commonsense reasoning capabilities of Large Language Models while preserving the formal reliability of symbolic solvers, an arrangement that could be particularly transformative for real world robotics, where environment fluctuations and concurrency constraints are significant.

Building on this context, our study systematically compares the planning capabilities of nine LLMs and Fast Downward across five distinct PDDL domains: *barman*, *blocks*, *elevator*, *satellite*, and *tidybot*. Each domain reflects varying degrees of complexity in resource management, concurrency, and object handling. We focus not only on raw success rates but also on execution fidelity, the extent to which generated plans translate into executable actions. Our results underscore the contrast between the guaranteed executability of symbolic planners in many domains and the strong yet inconsistent performance of LLMs as planners. While LLMs excel in simpler tasks, they struggle in domains that place heavy demands on resource coordination. These findings reinforce the notion that although LLM based approaches

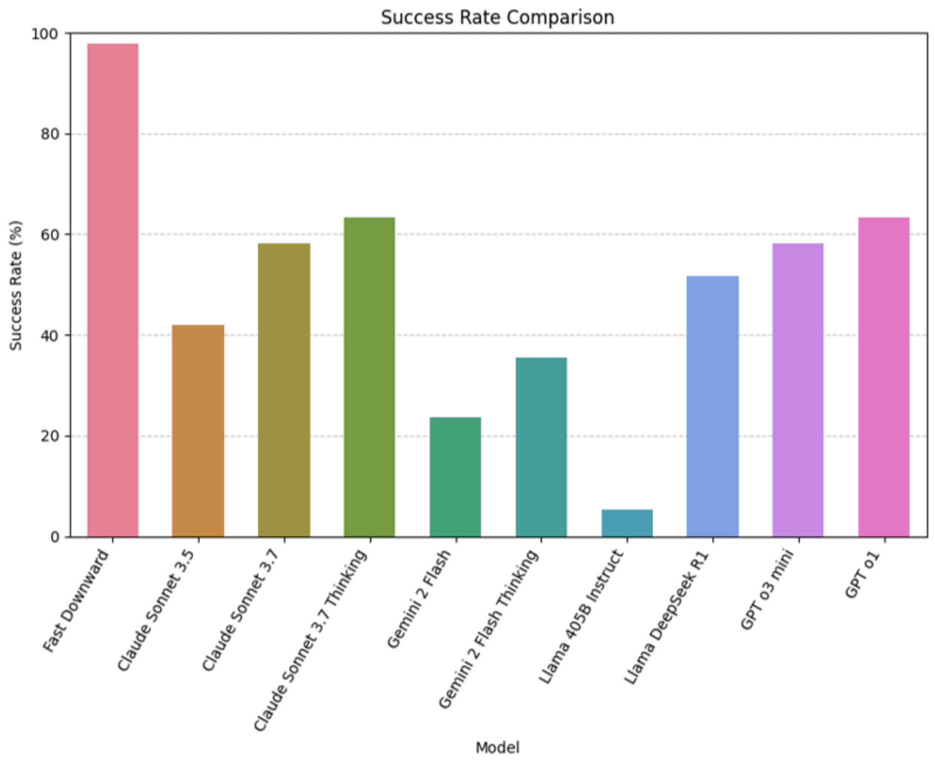

Figure 2: Comparison of the planning capabilities of nine different LLMs against the classical planner *Fast Downward*, which serves as the baseline. This figure presents the average success rate of solving the given problems across multiple PDDL domains. Among the LLM-based planners, *GPT-o1* and *Claude Sonnet 3.7 Thinking* achieve the highest success rate, both at 63.4%. Notably, the reasoning model *Claude Sonnet 3.7 Thinking* outperforms its non-reasoning counterpart *Claude Sonnet 3.7* (58.1%), and *Gemini 2 Flash Thinking* (35.5%) shows an improvement over *Gemini 2 Flash* (23.7%). However, even the best-performing models do not achieve robust planning. Further analysis, particularly in more complex domains, shows that LLMs still struggle, raising concerns about whether their success in well-known domains stems from genuine reasoning or merely from recalling solutions encountered during extensive pretraining. These findings highlight the need for plan validation and execution fidelity in LLM-driven planning for robotic tasks.

show considerable promise, they require reliable integration with classical planning frameworks to ensure consistent performance.

2 Related Work

Task planning has traditionally relied on symbolic methods, where planners generate optimal action sequences from well-defined representations, such as those defined in the Planning Domain Definition Language (PDDL). While methods like Fast Downward offer efficiency and formal verification, they struggle with uncertainty and require significant domain engineering. To mitigate these limitations, recent approaches explore LLM-based planning and hybrid methods that integrate symbolic reasoning with generative capabilities.

2.1 Emergence of LLM-Based Planning

2.1.1 Direct Use of LLMs as Planners

A growing body of research explores directly leveraging LLMs to generate multi step plans without explicit symbolic backends. In this approach, the LLM takes a task description as input and outputs an action sequence, often in natural language, for execution by an agent or robot. For instance, Huang *et al.* Huang et al. [2022] demonstrate “zero shot” action planning for embodied agents. Similarly, ReAct Yao et al. [2023] interleaves reasoning traces and actions within a single inference loop.

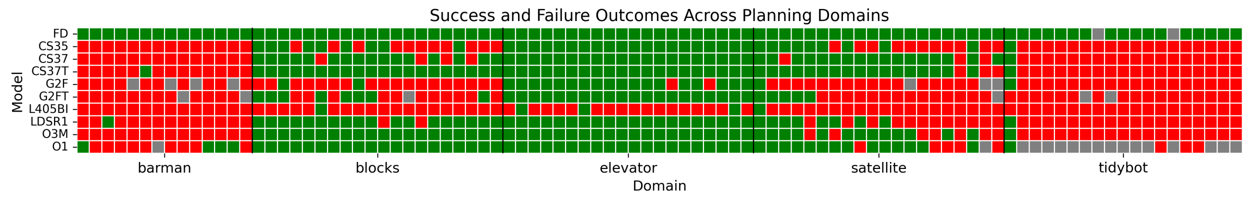

Figure 3: Per-domain success and failure outcomes of the evaluated LLMs compared to the baseline classical planner *Fast Downward* (FD). The evaluated models include CS35: *Claude Sonnet 3.5*, CS37: *Claude Sonnet 3.7*, CS37T: *Claude Sonnet 3.7 Thinking*, G2F: *Gemini 2 Flash*, G2FT: *Gemini 2 Flash Thinking*, L405BI: *Llama 405B Instruct*, LDSR1: *Llama DeepSeek R1*, O3M: *GPT-o3 mini*, and O1: *GPT-o1*. Success cases are visualized in green, failures in red, and cases where no plan was generated in gray. The five evaluated domains, *barman*, *blocks*, *elevator*, *satellite*, and *tidybot*, are each represented with their respective problem sets. In the *tidybot* domain, *Fast Downward* failed to find solutions for two problems. Overall, the best performing models are *Claude Sonnet 3.7 Thinking* and *GPT o1*. Notably, *GPT o1* sometimes concluded that no solution existed after reasoning, leading to no plan generation. In contrast, the two *Gemini* models failed to generate plans in some cases due to their limited maximum output tokens. Performance across domains varies: models perform well in the *blocks* and *elevator* domains, followed by the *satellite* domain, while all struggle with the *barman* and *tidybot* domains. It remains unclear whether success in certain domains results from genuine reasoning or if models are recalling solutions from their pretraining data. The high success rate of non-reasoning models in the *elevator* domain suggests that this domain may be covered in the pretraining corpus. In the *blocks* domain, improvements from newer reasoning models are evident, while the *barman* and *tidybot* domains remain challenging.

Though promising, purely LLM-driven planning has limitations. Chen *et al.* Chen et al. [2024] show that LLMs often fail to incorporate critical constraints in long-horizon tasks, while Cloos *et al.* Cloos et al. [2024] highlight that LLMs frequently overlook basic logical rules in puzzle-like environments. To mitigate these issues, Hee *et al.* Song et al. [2023] introduce LLM-Planner, embedding environment grounding and few-shot examples into the prompt. Recent frameworks, such as AgentGen Hu et al. [2025], further emphasize environment and task generation to enhance the diversity of LLM-based planning.

Overall, LLMs' ability to avoid hallucinated or contradictory steps remains a challenge. As shown by Kambhampati Kambhampati et al. [2024] and Valmeekam Valmeekam et al. [2023a], purely autoregressive outputs often lack robust self-verification mechanisms.

2.1.2 Hybrid PDDL + LLM Approaches

Another strand of research fuses PDDL-based planners with LLMs, aiming to leverage the strengths of both. Symbolic planners provide rigorous state-space search and correctness guarantees, while LLMs contribute commonsense reasoning and semantic understanding. For instance, Capitanelli and Mastrogiovanni Capitanelli and Mastrogiovanni [2024] propose Teriyaki, a neurosymbolic planner where *GPT-3* generates PDDL plans. Stein *et al.* Stein et al. [2024] automate prompt generation from PDDL domains, while Silver *et al.* Silver et al. [2024] show that *GPT-4* can internalize domain constraints to produce scalable plan generators in code form.

Further techniques emphasize iterative refinement and environment feedback. Mahdavi *et al.* Mahdavi et al. [2024] create feedback loops between an LLM-generated PDDL file and a running simulator. Zhou *et al.* Zhou et al. [2024] propose iterative self-refinement (ISR-LLM), incorporating a validator to revise generated plans. Other approaches strive to balance efficiency and accuracy. For instance, Kwon *et al.* Kwon et al. [2024] utilize neuro-symbolic goal decomposition to achieve this balance.

2.1.3 LLM-Based Task Planning in Robotics

LLMs are increasingly applied in robotics, where planning needs to consider physical constraints, sensor noise, and complex dynamics. Several studies demonstrate how LLMs can assist in or autonomously devise robotic action plans. Capitanelli and Mastrogiovanni Capitanelli and Mastrogiovanni [2024] extend GPT-3 with Teriyaki, while Joublin *et al.* Joublin et al. [2024] propose CoPAL, an iterative feedback method for real-time correction of robot action plans. Efforts such as SayCan Hazra et al. [2024] ground LLM-based reasoning with learned value functions, whereas PaLM-E Driess et al. [2023] integrates sensor input with textual prompts for multimodal embodied planning.

2.2 Benchmarks and Evaluation Metrics for LLM Planning

2.2.1 Classical Benchmarks (PDDL-Focused)

Efforts to systematically evaluate LLM-based planners often rely on classical IPC-style domains. PlanBench Valmeekam et al. [2023b] assesses LLMs on PDDL problems, highlighting common errors. A follow-up study by Valmeekam *et al.* Valmeekam et al. [2024] contrasts standard LLMs with "large reasoning models." The Planetarium framework Zuo et al. [2024] introduces automated checks for semantic equivalence in PDDL, while frameworks such as NL2Plan Gestrin et al. [2024] investigate text-to-PDDL conversion.

2.2.2 Everyday Tasks and Natural Language Benchmarks

Another research direction focuses on natural-language planning scenarios. NATURAL PLAN Zheng et al. [2024] evaluates LLMs on multi-constraint tasks, while TravelPlanner Chen et al. [2024] challenges LLMs with travel plans. Meanwhile, "Baba Is AI" Cloos et al. [2024] is a puzzle-like environment designed to test LLM reasoning.

2.2.3 Emerging Metrics

Traditional planning metrics include success rate, plan length, and runtime. However, execution fidelity, defined as the extent to which generated plans translate into executable actions, is also an important consideration. Open Grounded Planning Guo et al. [2024] tests text-based plans, while CAT-BENCH Lal et al. [2024] evaluates causal-temporal ordering.

3 Methods

This study examines the planning capabilities of various LLMs compared to Fast Downward, a well-established classical planner for task planning, using PDDL domain and problem descriptions. This section details the evaluation setup, the LLMs assessed, the domains selected for benchmarking, the prompts employed to generate plans, and the evaluation metrics utilized.

3.1 Fast Downward Baseline

Fast Downward is a domain-independent planning system that translates PDDL tasks into multi-valued variable representations, highlighting implicit constraints and dependencies. We use the `seq-sat-lama-2011` heuristic configuration for our experiments. This configuration integrates landmark and FF heuristics within an iterated search scheme, balancing speed and plan quality through a sequence of lazy greedy best-first searches and weighted `A*` steps. This anytime approach refines initial solutions, allowing the planner to improve plan quality if time remains. In our setup, we cap the planning time at 600 seconds, offering a robust classical benchmark against which LLM-based methods can be compared. By leveraging hierarchical decomposition and heuristic guidance, Fast Downward efficiently handles diverse domains, reinforcing its position as a reliable point of reference for evaluating emerging LLM-based planners.

3.2 Overview of evaluated LLMs

In our study, we evaluate the LLMs listed in Table 1. Given the rapid advancements in the field, this evaluation represents a snapshot of the current state of the art. We acknowledge the release of new models such as *Grok-3* and *DeepSeek-R1*, which were not included in our evaluation due to inaccessibility at the time of writing. To include the reasoning-optimized *DeepSeek-R1* model in our study, we used the *Llama DeepSeek R1* version, which is a distilled version of the original model. We accessed the models *Llama DeepSeek R1* and *Llama 405B Instruct* via the provider `fireworks.ai` Fireworks AI [2024], a platform that hosts and provides access to a variety of AI models.

To provide a reference point for the reasoning capabilities of the evaluated LLMs, Table 1 includes a brief description of each model's reasoning capabilities and their GPQA Diamond Accuracy Rein et al. [2023]. The GPQA Diamond benchmark consists of 448 challenging multiple-choice questions in biology, physics, and chemistry, designed to be difficult for both human experts and AI models. Higher accuracy on the GPQA Diamond benchmark indicates superior reasoning capabilities. The accuracy values in the table are collected from various sources Team [2025], Research [2025], Olteanu [2025], Sharma [2025], xAI [2025], Anthropic [2025].

Table 1: Evaluated LLMs

Model	Type	Reasoning Capabilities	Release Date	GPQA
Claude Sonnet 3.5	Proprietary	Advanced multi-step reasoning and code generation, demonstrating superior logical inference.	June 2024	59.4
Claude Sonnet 3.7	Proprietary	Hybrid reasoning model with improved structured problem-solving.	Feb 2025	68.0
Claude Sonnet 3.7 Thinking	Proprietary	Extended thinking mode enabling detailed, step-by-step reasoning for complex problem-solving.	Feb 2025	84.8
Gemini 2 Flash	Proprietary	General-purpose multimodal LLM optimized for efficiency but lacks explicit reasoning optimization.	Dec 2024	58.6
Gemini 2 Flash Thinking	Proprietary	Experimental variant specifically designed for enhanced reasoning and problem-solving tasks.	Jan 2025	74.2
Llama 405B Instruct	Open-Source	State-of-the-art open-source model with competitive reasoning performance.	July 2024	50.7
Llama DeepSeek R1	Open-Source	Optimized for reasoning via reinforcement learning with robust chain-of-thought solutions.	Jan 2025	71.5
GPT-o3 mini	Proprietary	Compact reasoning-optimized model trained for enhanced problem-solving.	Jan 2025	79.7
GPT-o1	Proprietary	Explicit chain-of-thought reasoning strategies.	Dec 2024	75.7

3.3 Domains Evaluated

These benchmarking domains cover a range of manipulation, scheduling, and resource-management challenges. Below, we summarize each domain's main features:

- **Barman (IPC 2014, 14 problems)**: Involves a robot bartender preparing and serving drinks using tools like dispensers and shakers. Each glass must be empty and clean before filling, and only one object can be held at a time, introducing tight sequencing constraints.
- **Blocks (IPC 2000, 20 problems)**: A classic typed STRIPS domain where a robot arm stacks blocks into a goal configuration. It tests careful action ordering, ensuring a block is clear before it can be moved, revealing combinatorial challenges.
- **Elevator (IPC 2000, 20 problems)**: Simulates transporting passengers across multiple floors. Planners must handle embarkation and disembarkation, optimizing routes to deliver passengers efficiently. Proper sequencing is required to avoid missed pickups and minimize travel time.
- **Satellite (IPC 2004, 20 problems)**: Focuses on coordinating multiple satellites to collect image data. Satellites must be calibrated and oriented correctly to capture targets. Scheduling transmissions and adjusting instruments within time windows adds resource-management complexity.
- **Tidybot (IPC 2011, 19 problems)**: Models a household cleaning task. One or more robots navigate a 2D grid, picking up items and placing them onto goal locations like tables or cupboards. Gripper reach is limited, and objects can obstruct each other, often requiring the use of carts to move multiple items efficiently.

3.4 LLM Prompt for Plan Generation

To ensure consistency and interpretability in LLM-generated plans, we design our prompt to encourage structured reasoning. The LLM first provides a high-level reasoning overview, outlining its strategy for solving the planning problem, followed by a step-by-step sequence of actions, each with an explicit justification.

3.4.1 Task Description

The LLM acts as a planning assistant that reads PDDL domain and PDDL problem descriptions, then generates a structured plan. The response must include:

- A high-level reasoning section describing the overall approach.

- A detailed plan, listed as a sequence of actions.

3.4.2 Input

- **PDDL domain:** Defines predicates, actions, and constraints.
- **PDDL problem:** Specifies objects, the initial state, and the goal conditions.

3.4.3 Output Format

The response must be valid JSON with two key components:

- **Reasoning:** A high-level explanation describing the approach.
- **Plan:** An ordered list of actions. Each action object includes:
  - *name*: Identifier of the action (matching the domain).
  - *parameters*: Objects the action applies to.
  - *reason*: Justification for choosing this action.
  - *confirm_reasoning*: A validation statement confirming correctness.

Below the template of the JSON response format:

```
{
  "reasoning": [
    "A high-level explanation of the overall plan, describing how it transitions from the initial
        state to the goal."
  ],
  "plan": [
    {
      "name": "action_name",
      "parameters": ["arg1", "arg2"],
      "reason": "Explanation of why this action was chosen.",
      "confirm_reasoning": "Final validation statement."
    }
  ]
}
```

3.5 Evaluation Metrics

The following metrics were employed to assess the planners:

- **Success Rate (SR)**: The fraction of problem instances successfully solved. This metric is essential for comparing the baseline capability of each planner. A solution is marked successful if the planner produces a valid plan that satisfies the domain's goals.
- **Plan Length (PL)**: Measures the number of actions (or steps) in a generated plan. Classical planners typically produce near-optimal plans or follow heuristic-driven strategies, whereas LLM-based plans tend to be more variable. A plan that is too short may omit necessary actions, leading to incomplete execution, while excessively long plans increase the likelihood of errors or execution failures.
- **Executed Actions (Ac)**: Indicates how many actions from a generated plan can be validly executed according to the domain's constraints. This highlights gaps between theoretical output and practical feasibility. LLM-based methods can struggle when the chain of actions is incomplete or violates preconditions.
- **Execution Fidelity**: The ratio of executed actions to planned actions, indicating how well the planner's output aligns with the actual execution. A high fidelity score suggests that the planner generates plans that are more likely to be executable in practice.
- **Planning Time**: The time taken to generate a plan. This metric is crucial for real-time applications, where rapid decision-making is essential. LLM-based planners may require more time to reason through complex problems, potentially limiting their applicability in time-sensitive scenarios.

Table 2: Average Planning Time per Model

LLM Model	**Avg. Planning Time (s)**
Claude Sonnet 3.5	14.22
Claude Sonnet 3.7	28.90
Claude Sonnet 3.7 Thinking	112.61
Gemini 2 Flash	15.13
Gemini 2 Flash Thinking	22.02
GPT-o1	140.51
GPT-o3 mini	98.61
Llama 405B Instruct	27.08
Llama DeepSeek R1	160.15

Table 3: Performance Metrics for Planning Systems Across Five Domains. SR: Success Rate (%), PL: Plan Length, Ac: Executed Actions

Model	**Barman**			**Blocks**			**Elevator**			**Satellite**			**Tidybot**			**MEAN**		
	SR	**PL**	**Ac**	**SR**	**PL**	**Ac**	**SR**	**PL**	**Ac**	**SR**	**PL**	**Ac**	**SR**	**PL**	**Ac**	**SR**	**PL**	**Ac**
Fast Downward	100.0	98.7	98.7	100.0	26.9	26.9	100.0	10.4	10.4	100.0	40.0	40.0	89.5	39.5	39.5	97.85	39.56	39.56
Claude Sonnet 3.5	0.0	14.6	3.9	45.0	19.6	14.1	**100.0**	10.3	10.3	45.0	23.6	23.1	**5.3**	14.8	2.7	41.94	16.69	11.34
Claude Sonnet 3.7	0.0	96.9	10.6	80.0	21.6	19.6	**100.0**	9.6	9.6	**85.0**	40.1	34.6	**5.3**	36.0	3.5	58.06	37.26	16.02
Claude Sonnet 3.7 T.	7.1	99.9	20.9	**100.0**	20.0	20.0	**100.0**	9.3	9.3	**85.0**	47.8	44.7	**5.3**	36.1	4.3	**63.44**	38.98	19.92
Gemini 2 Flash	0.0	41.8	1.4	10.0	20.5	8.8	90.0	14.5	13.9	5.0	31.0	3.2	**5.3**	34.6	2.4	23.66	27.53	6.28
Gemini 2 Flash T.	0.0	90.0	4.4	40.0	19.3	13.0	**100.0**	15.1	15.1	25.0	55.2	21.4	0.0	30.7	3.6	35.48	39.08	12.02
Llama 405B Instruct	0.0	48.9	4.1	5.0	17.4	6.0	15.0	9.7	1.4	5.0	38.1	3.9	0.0	20.1	2.5	5.38	25.46	3.55
Llama DeepSeek R1	7.1	54.9	12.4	90.0	18.4	17.2	**100.0**	10.1	10.1	40.0	32.1	25.4	**5.3**	17.2	3.3	51.61	24.77	13.86
GPT-o3 mini	0.0	95.9	7.6	**100.0**	18.5	18.5	**100.0**	9.7	9.7	65.0	65.0	46.7	**5.3**	34.3	6.3	58.06	41.48	18.53
GPT-o1	**28.6**	78.8	44.6	**100.0**	19.3	19.3	**100.0**	10.9	10.9	70.0	32.2	27.1	**5.3**	4.5	0.9	**63.44**	26.18	19.22

4 Results and Analysis

4.1 Success Rate & Overall Planner Performance

Our experiments reveal a progressive enhancement in LLM-based planners. The top-performing models, *Claude Sonnet 3.7 Thinking* and *GPT-o1*, each achieve a success rate of 63.4%, outperforming their non-reasoning counterparts, *Claude Sonnet 3.7* and *GPT-o3 mini* (Figure 2). This improvement suggests that incorporating reasoning mechanisms enhances planning capabilities. A detailed analysis of per-domain performance can be found in Table 3.

However, even the best-performing models do not achieve robust planning. Figure 3 shows that LLMs struggle in more complex domains, raising concerns about whether their success in familiar domains stems from genuine reasoning or simply recalling solutions from pretraining. The emergence of reasoning models, such as *Claude Sonnet 3.7 Thinking* and *Gemini 2 Flash Thinking*, demonstrates progress, yet further analysis confirms that LLMs still face challenges in complex domains. These findings highlight the necessity of plan validation and execution fidelity in LLM-driven planning for robotic tasks.

4.2 Plan Length, Executed Actions, and Execution Fidelity

The plan lengths generated by *Claude Sonnet 3.7*, *Claude Sonnet 3.7 Thinking*, *GPT-o3 mini*, and *Gemini 2 Flash Thinking* align most closely with the baseline established by the *Fast Downward* planner, which produces plans with an average length of 39.56 actions. Among these models, *GPT-o3 mini* is the only one that slightly overestimates plan lengths, generating an average of 41.48 actions per plan. In contrast, the others tend to underestimate plan lengths, with *Claude Sonnet 3.7 Thinking* at 38.98 actions and *Gemini 2 Flash Thinking* at 39.08 actions. *GPT o1* generates notably shorter plans, averaging 26.18 actions, which is largely due to its tendency to conclude that no solution exists in certain cases, leading to no plan generation (Figure 3). The shortest average plan length is observed in *Claude Sonnet 3.5*, which generates plans with only 16.69 actions, less than half of the *Fast Downward* baseline.

The executed actions metric evaluates how many actions from a generated plan can be validly executed within the domain's constraints. The models that perform best according to this metric are *Claude Sonnet 3.7 Thinking* and *GPT-o1*, which execute an average of 19.92 and 19.22 actions, respectively, before violating domain constraints. *GPT-o3 mini* also performs relatively well, executing 18.53 actions on average. In contrast, models with lower plan lengths tend to execute fewer steps. For example, *Claude Sonnet 3.5*, with an average plan length of 16.69 actions, successfully executes only 11.34 steps. The worst-performing model is *Llama 405B Instruct*, which executes just 3.55 actions on average, failing even in the *elevator* domain where all other models perform well.

Execution fidelity, defined as the ratio of executed actions to plan length, provides insight into how well a model adheres to domain constraints. As shown in Figure 1, *GPT-o1* achieves the highest execution fidelity at 73.4%, followed by *Llama DeepSeek R1* at 55.9% and *Claude Sonnet 3.5* at 51.1%. Notably, despite having the fourth-lowest success rate (41.94%), *Claude Sonnet 3.5* exhibits relatively high execution fidelity, meaning that while its plans are often valid in terms of execution, their ability to achieve the goal remains uncertain. This contrasts with models such as *Claude Sonnet 3.7 Thinking*, which has a higher success rate (63.44%) but a lower execution fidelity due to its longer plan length. The lowest execution fidelity is observed in *Llama 405B Instruct*, where only 13.9% of planned actions can be executed. This highlights the model's inability to generate valid plans and follow domain constraints effectively.

4.3 Planning Time

Planning time varies considerably across models, as shown in Table 2. The fastest model, *Claude Sonnet 3.5*, requires an average of 14.22 seconds per plan, while the slowest, *Llama DeepSeek R1*, takes 160.15 seconds. Generally, models with explicit reasoning, such as *Claude Sonnet 3.7 Thinking* (112.61 s) and *GPT-o1* (140.51 s), exhibit longer planning times compared to their non-reasoning counterparts. This aligns with the expectation that reasoning increases computational overhead. There is no strict correlation between planning time and success rate. While *GPT-o1* (140.51 s) achieves the highest success rate (63.44%), *Llama DeepSeek R1* (160.15 s) has a lower success rate (51.61%) despite requiring more time. Conversely, some faster models, such as *Claude Sonnet 3.7* (28.90 s), outperform slower ones like *GPT-o3 mini* (98.61 s), indicating that longer planning times do not necessarily result in better plans. It is important to note that planning time comparisons are not entirely fair, as response times can fluctuate due to varying server loads. Models hosted on heavily used services may experience temporary slowdowns. Additionally, *Llama 405B Instruct* and *Llama DeepSeek R1* are accessed via the `Fireworks.ai` provider rather than an official hosting service, which may introduce additional latency.

4.4 Domain-Specific Insights

The performance of LLM-based planners varies across domains (Figure 3). Models perform well in the *blocks* and *elevator* domains, where the problem structures align well with patterns they have likely encountered during training. In contrast, the *barman* and *tidybot* domains remain challenging due to their complex action dependencies and constraints. The consistently high success rates in the *elevator* domain suggest that it may be well-represented in the models' pretraining data. Table 3 illustrates how reasoning models improve execution fidelity; however, performance gaps remain in domains that require more elaborate planning strategies.

5 Discussion, Conclusion & Future Work

Robust planning is essential for autonomous robotic systems, where execution failures can lead to critical errors. Our study evaluates whether LLMs can function as general planners, similar to classical solvers like *Fast Downward*, given a domain and problem definition. While models such as *GPT-o1* and *GPT-o3 mini* demonstrate promising success rates and execution fidelity, they still struggle with strict domain constraints, making them unreliable for real-world robotic applications. Execution failures, even partial ones, highlight the difficulty of ensuring action validity over long planning horizons.

Future research should explore hybrid approaches that integrate LLMs with classical planning techniques, combining LLM flexibility with symbolic verification to ensure robustness. Additionally, iterative plan refinement, where an LLM continuously adjusts a plan as execution unfolds, could better align with how humans approach long horizon tasks. If LLMs can truly reason about PDDL structures and maintain internal state consistency, they may serve as adaptive planning components within autonomous robotic systems. While LLM based planning remains immature, its rapid progress suggests potential for new learning driven planning architectures that blend data driven reasoning with structured robotic task execution.

References

Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006. doi:10.1613/jair.1705.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL—the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. doi:10.1109/ICCV.2023.2998.

Maitrey Gramopadhye and Daniel Szafir. Generating Executable Action Plans with Environmentally-Aware Language Models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3568–3575, 2023. doi:10.1109/IROS56739.2023.10045678.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, and et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 3–14, 2022. doi:10.48550/arXiv.2204.01691.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B. Murthy. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Subbarao Kambhampati. Can Large Language Models Reason and Plan? *Annals of the New York Academy of Sciences*, 2024. doi:10.1111/nyas.15125.

Sadegh Mahdavi, Raquel Aoki, Keyi Tang, and Yanshuai Cao. Leveraging environment interaction for automated pddl translation and planning with large language models. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

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.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147, 2022.

Yanan Chen, Ali Pesaranghader, Tanmana Sadhu, and Dong Hoon Yi. Can We Rely on LLM Agents to Draft Long-Horizon Plans? Let's Take TravelPlanner as an Example. arXiv preprint arXiv:2408.06318, 2024.

Nathan Cloos, Meagan Jens, Michelangelo Naim, Yen-Ling Kuo, Ignacio Cases, Andrei Barbu, and Christopher J. Cueva. Baba Is AI: Break the Rules to Beat the Benchmark. In *Proceedings of the ICML 2024 Workshop on LLMs and Cognition*, 2024.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. AgentGen: Enhancing Planning Abilities for Large Language Model based Agent via Environment and Task Generation. arXiv preprint arXiv:2408.00764, 2025. Accepted by KDD 2025 (Research Track).

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models: A critical investigation. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023a.

Alessio Capitanelli and Fulvio Mastrogiovanni. A Framework for Neurosymbolic Robot Action Planning using Large Language Models. *Frontiers in Neurorobotics*, 18, 2024. doi:10.3389/fnbot.2024.1342786.

Katharina Stein, Daniel Fišer, Jörg Hoffmann, and Alexander Koller. Automating the Generation of Prompts for LLM-based Action Choice in PDDL Planning. In *Proceedings of the ICAPS 2024 Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL)*, 2024. Preprint available at arXiv:2311.09830.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized Planning in PDDL Domains with Pretrained Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18), 2024. doi:10.1609/aaai.v38i18.30006.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei 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)*, pages 2081–2086, 2024. doi:10.1109/ICRA57147.2024.10610065.

Minseo Kwon, Yaesol Kim, and Young J. Kim. Fast and Accurate Task Planning using Neuro-Symbolic Language Models and Multi-level Goal Decomposition. arXiv preprint arXiv:2409.19250, 2024.

Frank Joublin, Antonello Ceravola, Pavel Smirnov, Felix Ocker, Joerg Deigmoeller, Anna Belardinelli, Chao Wang, Stephan Hasler, Daniel Tanneberg, and Michael Gienger. CoPAL: Corrective Planning of Robot Actions with Large Language Models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8664–8670. IEEE, 2024. doi:10.1109/ICRA57147.2024.10610434.

Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. Saycanpay: Heuristic planning with large language models using learnable domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20123–20133, 2024. doi:10.1609/aaai.v38i18.29991.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, and et al. PaLM-E: An Embodied Multimodal Language Model. arXiv preprint arXiv:2303.03378, 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*, 36:38975–38987, 2023b.

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, 2024.

Max Zuo, Francisco Piedrahita Velez, Xiaochen Li, Michael L. Littman, and Stephen H. Bach. Planetarium: A Rigorous Benchmark for Translating Text to Structured Planning Languages. arXiv preprint arXiv:2407.03321, 2024.

Elliot Gestrin, Marco Kuhlmann, and Jendrik Seipp. NL2Plan: Robust LLM-Driven Planning from Minimal Text Descriptions. arXiv preprint arXiv:2405.04215, 2024.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. NATURAL PLAN: Benchmarking LLMs on Natural Language Planning. arXiv preprint arXiv:2406.04520, 2024.

Shiguang Guo, Ziliang Deng, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. Open Grounded Planning: Challenges and Benchmark Construction. arXiv preprint arXiv:2406.02903, 2024.

Yash Kumar Lal, Vanya Cohen, Nathanael Chambers, Niranjan Balasubramanian, and Ray Mooney. CaT-Bench: Benchmarking Language Model Understanding of Causal and Temporal Dependencies in Plans. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19336–19354, Miami, Florida, USA, 2024. doi:10.18653/v1/2024.emnlp-main.1077.

Fireworks AI. The production ai platform built for developers, 2024. URL `https://fireworks.ai/`. Accessed: 2025-02-27.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

DataCamp AI Research Team. Deepseek vs. openai: Comparing the new ai titans. *DataCamp Blog*, 2025. URL `https://www.datacamp.com/blog/deepseek-vs-openai`. Accessed: 2025-02-27.

IBM AI Research. Meta releases new llama 3.1 models, including highly anticipated 405b parameter variant. *IBM Newsroom*, 2025. URL `https://www.ibm.com/think/news/meta-releases-llama-3-1-models-405b-parameter-variant`. Accessed: 2025-02-27.

Alex Olteanu. Gemini 2.0 flash thinking experimental: A guide with examples. *DataCamp Blog*, February 2025. URL `https://www.datacamp.com/blog/gemini-2-0-flash-experimental`. Accessed: 2025-02-27.

Swarit Sharma. Comparing claude 3.7 sonnet, claude 3.5 sonnet, openai o3-mini, deepseek r1, and grok 3 beta. *Passionfruit Blog*, February 2025. URL `https://www.getpassionfruit.com/blog/comparing-claude-3-7-sonnet-claude-3-5-sonnet-openai-o3-mini-deepseek-r1-and-grok-3-beta`. Accessed: 2025-02-27.

xAI. Grok 3 beta—the age of reasoning agents. *xAI Blog*, February 2025. URL `https://x.ai/blog/grok-3`. Accessed: 2025-02-27.

Anthropic. Claude 3.7 sonnet and claude code. *Anthropic News*, February 2025. URL `https://www.anthropic.com/news/claude-3-7-sonnet`. Accessed: 2025-02-27.