# PlanGenLLMs: A Modern Survey of LLM Planning Capabilities

**Hui Wei,**[†] **Zihao Zhang,**[‡] **Shenghua He,**[§] **Tian Xia,**[§]
**Shijia Pan,**[†] **Fei Liu**[‡]

[†]University of California, Merced [‡]Emory University [§]PAII Inc.
{huiwei2, span24}@ucmerced.edu
{zihao.zhang, fei.liu}@emory.edu {shenghh2015, TianXia0209}@gmail.com

## Abstract

LLMs have immense potential for generating plans, transforming an initial world state into a desired goal state. A large body of research has explored the use of LLMs for various planning tasks, from web navigation to travel planning and database querying. However, many of these systems are tailored to specific problems, making it challenging to compare them or determine the best approach for new tasks. There is also a lack of clear and consistent evaluation criteria. Our survey aims to offer a comprehensive overview of current LLM planners to fill this gap. It builds on foundational work by Kartam and Wilkins (1990) and examines six key performance criteria: completeness, executability, optimality, representation, generalization, and efficiency. For each, we provide a thorough analysis of representative works and highlight their strengths and weaknesses. Our paper also identifies crucial future directions, making it a valuable resource for both practitioners and newcomers interested in leveraging LLM planning to support agentic workflows.[1]

## 1 Introduction

Planning, which involves generating a sequence of actions to reach a desired goal state (Newell et al., 1958; Kartam and Wilkins, 1990), is fundamental to human intelligence. For example, when planning a trip to San Francisco, one would search for flights, book tickets based on budget and schedule, arrange local transportation to the airport, and consider alternatives in case of cancellations. These planning tasks require complex reasoning, world knowledge, decision-making, and the ability to adapt, making them a significant challenge for humans. To date, there has been a growing focus on developing LLM planners to automate these complex tasks.

A comprehensive survey of LLM planners would significantly propel research in this field. Prior studies have explored planning methods and evaluation benchmarks (Huang et al., 2024c; Li et al., 2024d). Huang et al. (2024c) categorized planning methods into decomposition, plan selection, external modules, reflection, and memory, while Li et al. (2024d) reviewed evaluation benchmarks across various domains. However, many of these benchmarks and systems are tailored to specific problems, making it hard to compare LLM planners across domains or determine the best planner for new tasks. Further, there is a lack of clear and consistent evaluation criteria. We believe this gap may hinder the development of advanced LLM planners.

Our survey builds on the foundational work of Kartam and Wilkins (1990) to address key evaluation criteria for LLM planners. The original paper highlighted challenges in evaluating early AI planning systems, which relied on heuristics and were confined to research labs. The initial criteria were categorized into performance, representation, and communication issues. With more advanced LLM planning, we reexamine this critical framework and focus on six key evaluation criteria: *completeness*, *executability*, *optimality*, *representation*, *generalization*, and *efficiency*. For each criterion, we provide a thorough analysis of representative works, highlighting their strengths and weaknesses.

We contribute to the literature by addressing key research questions in LLM planning: What foundational capabilities distinguish them from earlier AI planners? How can we comprehensively measure their performance? We examine the datasets, evaluation methods, and metrics available to the community. We also highlight crucial areas where research is still lacking, including representation, hallucination, alignment, multi-agent planning, connections to agentic workflows, aiming to fill these gaps and advance the field. Figure 1 presents a taxonomy of six key performance criteria and representative

[1]A curated list of papers and resources related to this survey are available at https://github.com/wll199566/Awesome-LLM-Planning-Capability.

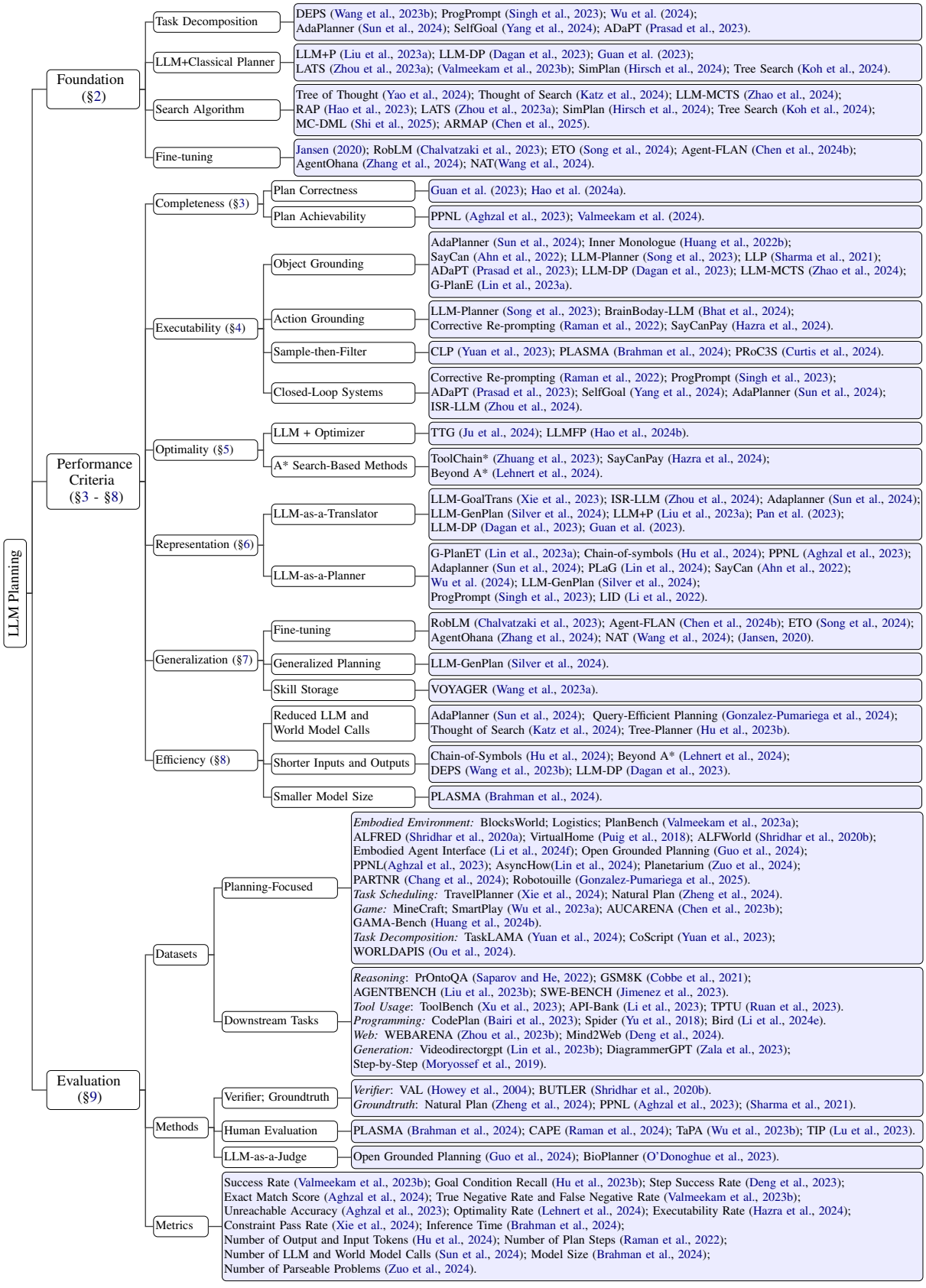


Figure 1: Taxonomy of LLM Planning

techniques. For those new to LLM planning, we recommend a thorough read, while experts can focus on specific sections. Each section offers clear definitions, relevant works, and includes links to tables in the Appendix. We will dive into the details in the following sections.

## 2 LLM Planning Foundations (Tables 1-3)

We begin by exploring LLM planning foundations, covering widely-used paradigms to provide background for readers unfamiliar with the field. It is broken down into four parts.

**Task Decomposition** Task decomposition breaks down abstract goals into specific, manageable subgoals. It helps mitigate errors by enabling verification at each step and makes LLM reasoning more tractable by narrowing the knowledge space.

Task decomposition can be performed *sequentially, in parallel, or asynchronously*. Specifically, sequential decomposition (Wang et al., 2023b; Singh et al., 2023; Sun et al., 2024; Wu et al., 2024) requires that the precondition of the subsequent subgoal is the effect of the preceding subgoal. In contrast, parallel decomposition (Yang et al., 2024) involves subgoals that share the *same* precondition and effect, where achieving the final goal requires completing only *one* of these subgoals. Asynchronous decomposition (Lin et al., 2024) involves parallelizing subgoals as well. However, these subgoals in distinct branches have *unique* preconditions and effects. Asynchronous decomposition requires the completion of *all* subgoals to achieve the overall goal.

Moreover, task decomposition can be performed *recursively*, applying any of the above three approaches at each step. For example, Prasad et al. (2023) recursively break down the goal until each subgoal can be easily executed in the environment.

**LLM + Classical Planner** Studies (Valmeekam et al., 2023b; Kambhampati, 2024; Kambhampati et al., 2024) show that LLMs struggle with independent planning. Classical planners, such as Fast Downward (Helmert, 2006), ensure correct plans but depend on experts to translate user queries into formal representations, limiting scalability. A hybrid approach integrating LLMs with classical planners combines the world knowledge of LLMs with the precision and reliability of classical methods, addressing their individual limitations.

When integrated with classical planners, LLMs *translate natural language problems into formal representations* or *generate initial plans*. For example, LLM+P (Liu et al., 2023a), LLM-DP (Dagan et al., 2023), and Guan et al. (2023) use LLMs to convert planning problems into PDDL (McDermott et al., 1998), solved by Fast Downward or BFS(f) (Lipovetzky et al., 2014). Valmeekam et al. (2023b) employs LLMs to generate an initial plan, guiding the LPG planner (Gerevini et al., 2002), which iteratively refines it until a correct solution is found.

**Search Algorithm** Search algorithms, including *Breadth-First Search, Depth-First Search* (Yao et al., 2024; Katz et al., 2024), *Monte Carlo Tree Search* (Hao et al., 2023; Zhou et al., 2023a; Zhao et al., 2024; Shi et al., 2025), and *Greedy Best-First Search* (Koh et al., 2024; Hirsch et al., 2024), have been applied to improve LLM-based planning. These algorithms treat planning as a search problem, using search policy to guide the exploration of various possibilities. Search algorithms excel in planning problems by offering systematic exploration, optimality guarantees, and formal verification without requiring extensive domain knowledge, though they may be computationally intensive compared to more specialized methods like task decomposition.

All search algorithms consist of four core components: (1) *Search Policy* determines node exploration order, which are defined by the underlying search algorithm and are independent of LLMs. (2) *Expansion* generates possible actions from a state, often using LLMs to propose actions based on user instructions and current environment. (3) *World Models* define state transitions based on action preconditions and effects, using LLMs (Hao et al., 2023), classical planners (Hirsch et al., 2024), or external environment simulators (Zhou et al., 2023a; Zhao et al., 2024; Koh et al., 2024). (4) *Evaluation* assesses state progress toward the goal via scores computed by predefined functions (Katz et al., 2024), LLM/LVM ratings (Yao et al., 2024; Hao et al., 2023; Zhou et al., 2023a), log-likelihood scores (Hirsch et al., 2024), voting (Yao et al., 2024), self-consistency scores (Zhou et al., 2023a) or reward models (Chen et al., 2025).

**Fine-tuning** Commonly used pretrained LLMs (e.g., GPT-4 (Achiam et al., 2023), Claude 3 (Anthropic, 2024), Llama 2 (Touvron et al., 2023)) are not specifically trained for planning tasks, and prompt-based methods, which do not update model parameters, cannot fundamentally improve performance in these areas (Chen et al., 2023a;

Wang et al., 2024). Fine-tuning, either focused on *planning-specific tasks* or *broader agentic capabilities*, enhances planning correctness by directly updating LLM parameters.

Planning-specific fine-tuning involves training a pretrained model on planning-focused tasks (e.g., Blocksworld or ALFWorld (Shridhar et al., 2020b)), to improve planning performance. For example, Jansen (2020) and Chalvatzaki et al. (2023) fine-tuned GPT-2 (Radford et al., 2019) on ALFWorld, demonstrating its effectiveness in robotics planning.

Generalized agentic fine-tuning optimizes models using datasets that include both general tasks (e.g., question answering) and diverse agentic tasks (e.g., reasoning, planning, and tool use). This approach is motivated by two key insights: (1) focusing too narrowly on planning may degrade general capabilities (Chen et al., 2024b), and (2) agentic tasks share overlapping capabilities. For instance, reasoning and tool-use tasks often involve planning components. Thus, fine-tuning on a broader set of agentic tasks can simultaneously enhance planning performance and other interrelated capabilities, like reasoning, which are integral to planning. Moreover, standardizing trajectory formats from different tasks (Zhang et al., 2024; Chen et al., 2024b), as well as incorporating unsuccessful reasoning or planning trajectories (Wang et al., 2024; Chen et al., 2024b; Song et al., 2024) can further enhance learning and performance.

## 3 Criterion I: Completeness (Table 4)

The completeness of planning has two key aspects: (1) if a valid plan exists, the model should generate it correctly, and (2) if no feasible plan is possible, the model should recognize this and refrain from generating an incorrect or arbitrary plan.

A plan is correct if it achieves the goal within a fixed budget while avoiding excessive complexity and infinite loops (e.g., finishing booking a flight ticket to San Francisco within five hours meets the criteria for correctness). To ensure correctness, the LLM must work with classical sound and complete solvers (Guan et al., 2023; Hao et al., 2024a). Also, the LLM has to accurately translate the domain and problem into the specific format (e.g., PDDL), required by these solvers (Guan et al., 2023).

In terms of identifying unsolvable planning problems, those with inherently unachievable goals, even top LLMs (e.g., GPT-4 (Achiam et al., 2023)) and Large Reasoning Models (e.g., OpenAI O1 (Jaech et al., 2024)) struggle due to hallucination issues (Aghzal et al., 2023; Valmeekam et al., 2024; Katz et al., 2024).

## 4 Criterion II: Executability (Tables 5-6)

Executability checks if a plan can be carried out in a given environment while meeting all constraints. A executable plan must use only allowed actions and recognizable objects. Beyond basic precondition and postcondition rules, planners must consider extra constraints, such as avoiding sugar when baking a cake for diabetics (Yuan et al., 2023). Importantly, executability and correctness are orthogonal: *an executable plan isn't necessarily correct*, since it might be grounded and follow all constraints but still fail to reach the goal; likewise, *a correct plan isn't always executable* since it may only include high-level steps that can't be executed in a specific environment. Real-world applications typically require plans that are both correct and executable, especially when the executors are not humans (e.g., robots and computers).

To ensure plans are executable, researchers have proposed several approaches, including Object Grounding, Action Grounding, Closed-Loop Systems, and Sample-then-Filter.

**Object Grounding** Object grounding ensures the LLM planner uses objects available in the current environment when generating plans. E.g., if a stove is present but a microwave is not, the planner should correctly select the stove to heat a pancake and exclude the unavailable microwave from its plan. The simplest way to do this is by feeding observed or available objects into the planner via prompts (Huang et al., 2022b; Song et al., 2023; Lin et al., 2023a; Singh et al., 2023) or neural embeddings (Sharma et al., 2021; Ahn et al., 2022). In partially observed environments, where some object information are uncertain (e.g., needing to clean a cup that could be in a cabinet, drawer, or fridge), the planner can generate multiple possible plans, one for each scenario, and select the first feasible one (Prasad et al., 2023; Dagan et al., 2023; Zhao et al., 2024). Sun et al. (2024) takes a different approach, first generating a plan with placeholders for objects, then filling in the blanks with observed objects during execution.

**Action Grounding** Action grounding ensures all actions in a plan can actually be executed in the current environment. E.g., the planner should not

include an action like "draw a star" if the robot arm cannot perform such a complex operation. Instead, the task should be decomposed into simpler supported actions, such as multiple steps of "draw a line". Like object grounding, the simplest way is to explicitly list all admissible actions in LLMs' inputs (Singh et al., 2023). If a step goes beyond the executor's capabilities (e.g., combining multiple allowed actions into one), the LLM planner should be reprompted to break it down until every step is executable (Prasad et al., 2023).

Hierarchical Planning is another common method for grounding actions in LLM planning (Huang et al., 2022a; Raman et al., 2022; Song et al., 2023; Hazra et al., 2024; Bhat et al., 2024). It starts with high-level steps and then translates each one into a sequence of executable actions. This can be done in two ways: either generating all high-level steps first and then refining them into actions or translating each step as it's generated. If an action isn't exactly admissible, the closest valid action is retrieved instead (Huang et al., 2022a; Raman et al., 2022).

**Sample-then-Filter** Since LLMs alone can't guarantee plans meet all constraints, this approach first generates multiple plans and then verifies them, selecting only those that pass all checks. Yuan et al. (2023) ranks InstructGPT-generated plans using cosine similarity with task embeddings and selects the most similar one. Brahman et al. (2024) applies a verifier-guided beam search, keeping the top-K plans based on correctness and constraint adherence at each step. Curtis et al. (2024) generates Pythonic plans with parameter ranges, tests them with a simulator or classifier, and prompts the LLM to revise if constraints are still violated.

**Closed-Loop Systems** A closed-loop system in LLM planning means the model adapts its plan based on feedback from executors (Prasad et al., 2023; Yang et al., 2024), simulators (Bhat et al., 2024), validators (Zhou et al., 2024; Silver et al., 2024), other LLMs (Wang et al., 2023b; Zhou et al., 2023a), or even humans (Huang et al., 2022b), when the initial plan are inexecutable. It reprompts the LLM planner to replan until the plan is fully executable. Unlike open-loop systems (Huang et al., 2022a), which lack feedback, closed-loop planning helps reduce hallucinations and enables LLMs to handle complex, long-horizon, and dynamic environments (Wang et al., 2023b).

Closed-loop systems fall into two types: *implicit* and *explicit* (Sun et al., 2024). Implicit systems only fix the failed action (Raman et al., 2022; Singh et al., 2023; Zhou et al., 2024; Prasad et al., 2023; Yang et al., 2024), while explicit systems regenerate the entire plan (Sun et al., 2024). Though explicit systems require more computation, they prevent errors from compounding across steps.

## 5 Criterion III: Optimality (Table 7)

Optimality means achieving the goal state through the *best* possible plan. It poses a greater challenge than standard planning, which only requires reaching the goal state. Researchers have proposed two paradigms for achieving the optimal plans: LLM + Optimizer and $\mathbf{A}^*$ search-based methods.

**LLM + Optimizer** It combines the LLM, which turns user requests into symbolic optimization problems, with an optimizer that solves them and finds the best solution (Ju et al., 2024; Hao et al., 2024b). For example, TTG (Ju et al., 2024) uses the LLM to convert travel planning requests of minimum total costs into Mixed Integer Linear Programming problems, then runs an optimizer such as SCIP (Bestuzheva et al., 2021) to provide the optimal plan. Compared to LLM + classical planners, where LLMs define the domain and problem in a formal representation (e.g., PDDL), LLM + optimizers ensure optimal solutions by leveraging LLMs to formulate constrained optimization problems and classical optimizers to solve them. Unlike classical planners, which typically rely on search algorithms, heuristics and logical deductions and may not guarantee optimality (Russell and Norvig, 2016), optimizers, often using gradient-based methods (e.g., Newton's methods), can *guarantee optimal solutions* (Boyd and Vandenberghe, 2004).

**A* Search-Based Methods** A* search finds the lowest-cost optimal solution, particularly when using admissible heuristics that do not overestimate the actual cost to the goal. This makes it a natural choice for LLM-based planners to achieve optimality. ToolChain* (Zhuang et al., 2023) combines A* tree search with an LLM, which suggests next steps and estimates heuristic scores, to create plans with the fewest tool API calls. SayCanPay (Hazra et al., 2024) uses A* search with LLMs to generate the shortest possible plans. Beyond A* (Lehnert et al., 2024) trains a Transformer model, Searchformer, to mimic A* search paths for complex tasks like Maze navigation and Sokoban puzzles, optimizing for the fewest steps. Besides A* search, other

search algorithms (e.g., DFS and MCTS) could also be used to find optimal solutions, although without guarantee.

## 6 Criterion IV: Representation (Tab. 8-9)

In LLM planning, representation refers to how inputs and outputs are formatted. Inputs include domains (predicates and actions), problems (initial and goal states), and environmental observations, while outputs are the generated plans. Effective representation enhances problem comprehension and execution efficiency, especially given LLMs' sensitivity to prompts. We discuss this in two contexts: LLM-as-a-Translator and LLM-as-a-Planner.

**LLM-as-a-Translator** LLM-as-a-Translator converts between natural language (NL) and formal planning languages (e.g. PDDL), making classic planners more accessible to non-experts. By converting natural language tasks into formal representations and translating the resulting plans back into NL, LLMs reduce ambiguity, minimize hallucinations, and enable external validation, improving both usability and reliability in planning systems (Xie et al., 2023; Zhou et al., 2024; Sun et al., 2024; Silver et al., 2024).

Recent work has used LLMs to translate natural language descriptions into PDDL (Liu et al., 2023a; Guan et al., 2023; Xie et al., 2023; Dagan et al., 2023; Zhou et al., 2024; Tantakoun et al., 2025), LTL (Pan et al., 2023), and STL (Chen et al., 2024a). To ensure reliability, translations should be tested on development or external datasets like Planetarium (Zuo et al., 2024). If there are syntax or semantic errors, validators (e.g. VAL (Howey et al., 2004)) or human experts can provide feedback for the LLM to fix them.

**LLM-as-a-Planner** When LLMs act as standalone planners without classical planners or optimizers, various methods help encode environmental information, domains, and plans beyond just natural language. Environment and domain details have been represented using *tables* (Lin et al., 2023a), *condensed symbols* (Hu et al., 2024), *Pythonic code* (Aghzal et al., 2023; Singh et al., 2023; Sun et al., 2024), *neural embeddings* (Li et al., 2022; Ahn et al., 2022), and *graphs* (Lin et al., 2024; Wu et al., 2024). For generated plans, Pythonic code is a common alternative to natural language (Singh et al., 2023; Silver et al., 2024).

## 7 Criterion V: Generalization (Table 10)

Generalization refers to LLM planners' ability to apply learned strategies to new, more complex out-of-domain scenarios beyond its training environment, which can be enhanced through three key approaches: *fine-tuning* (described previously in Section 2), *generalized planning*, and *skill storage*. Given the diverse user queries in the real-world deployments, ensuring LLM planners' generalizability is important alongside other performance.

**Generalized Planning** Generalized planning extracts common patterns from a limited set of training solutions (i.e., plans) to solve unseen tasks within the same domain, which may be larger and more complex than the training tasks (Srivastava et al., 2011). For example, in the Delivery dataset (Yang et al., 2022), models trained on small-scale deliveries (9–17 locations) can generalize to larger ones (70–100 locations) using the same core strategy. Silver et al. (2024) approached this by prompting LLMs to summarize the domain and generate a minimal, generalizable Python-based plan.

**Skill Storage** Skill storage focuses on learning and reusing previously acquired skills to tackle new problems. E.g., Wang et al. (2023a) introduced a skill library that stores successfully executed skills (e.g., Combat Zombie). These skills are abstracted and generalized for reuse in similar situations (e.g., fighting spiders involves similar actions to fighting zombies). When encountering an unseen task, the LLM planning system retrieves relevant learned skills based on the task and current states, then applies them to generate an effective solution.

## 8 Criterion VI: Efficiency (Table 11)

Efficiency in LLM planning means reducing computational and monetary costs by decreasing LLM calls, world model interactions, input and output lengths, and model sizes. This is crucial especially developing planners based on commercial LLMs.

**Reduced LLM and World Model Calls** To decrease LLM and world model calls, several tricks are used: (1) generating the entire plan in one shot instead of step-by-step to reduce redundant prompts (Hu et al., 2023b; Sun et al., 2024; Gonzalez-Pumariega et al., 2024); (2) checking feasibility by world models only at the end of each subgoal, not after every action (Sun et al., 2024; Gonzalez-Pumariega et al., 2024); (3) merging plans with the same prefix actions or subgoals

to avoid duplicate world model checks when sampling multiple plans (Hu et al., 2023b); and (4) in tree search-based methods, querying the LLM once to generate a successor function and a goal check function. The successor function, which can produce all possible actions and states, provides state transitions based on the current state and selected action. The goal-check function determines whether the current state is the desired final goal. This approach avoids repeated LLM and world model calls at each node (Katz et al., 2024).

**Shorter Inputs and Outputs** Reducing input and output length includes *decreasing prompt and plan tokens* and *minimizing actions in the final plan* to alleviate the load on executors. For spatial reasoning and planning, (Hu et al., 2024) introduces Chain-of-Symbols (CoS), a compact symbolic representation that replaces natural language descriptions in CoT (Wei et al., 2022) trajectories. Lehnert et al. (2024) uses search dynamic bootstrapping to iteratively fine-tune a LLM, replacing training cases with solutions with less tokens and equal optimality. To minimize actions, Dagan et al. (2023) and Wang et al. (2023b) use action selectors based on predefined rules or trained models to find the shortest successful plan.

**Smaller Model Sizes** Shrinking the model size can reduce the computational burden, accelerating training and inference while lowering costs. To train a smaller planning model, Brahman et al. (2024) uses GPT-3 (Brown et al., 2020) as the teacher and T5 (Raffel et al., 2020) as the student, distilling the teacher's planning capabilities into the more compact student model.

## 9 Evaluation

**Datasets** LLM planning evaluation is conducted on two types of datasets: *planning-focused datasets* and *downstream-task datasets*.

Planning-focused datasets primarily assess planning abilities. The most common scenarios include (1) *Embodied environments*, (2) *Task scheduling* , (3) *Games*, and (4) *Task decomposition* (Li et al., 2024d). Figure 1 presents commonly used planning datasets; readers can refer to Li et al. (2024d, 2025) for further details.

While most of the datasets mentioned above assess whether the generated plans are correct, some specifically target key performance criteria in LLM planning. For *grounding*, Open Grounded Planning (Guo et al., 2024) and Embodied Agent Interface (Li et al., 2024f) evaluate performance in embodied environments, while CoScript (Yuan et al., 2023), TravelPlanner (Xie et al., 2023), and PPNL (Aghzal et al., 2023) focus on planning problems with constraints. For *representation*, Planetarium (Zuo et al., 2024) assesses LLMs' ability to translate natural language into PDDL. For *optimality*, Lin et al. (2024) and Gonzalez-Pumariega et al. (2025) introduce tasks requiring optimal plans using asynchronous actions. PPNL (Aghzal et al., 2023) can also evaluate a planner's ability to *identify unachievable goals (i.e., completeness)*.

Planning abilities can also be evaluated through downstream tasks, where planning is integral to task completion, and stronger planning skills enhance overall performance. Downstream tasks can be categorized as follows: (1) *Agentic tasks*, including reasoning-oriented tasks, tool-use-oriented tasks, programming tasks, and web tasks (Yu et al., 2018; Cobbe et al., 2021; Saparov and He, 2022; Zhou et al., 2023b; Deng et al., 2023; Liu et al., 2023b; Li et al., 2023; Xu et al., 2023; Jimenez et al., 2023; Ruan et al., 2023; Bairi et al., 2023; Li et al., 2024e), (2) *Generation tasks*, including video (Lin et al., 2023b), image (Zala et al., 2023) and text generation (Moryossef et al., 2019). Please refer to Figure 1 for example datasets.

**Methods** The most common approach to evaluating LLM planning is to test it in a simulated environment and validate the generated plans using either *an internal verifier* provided by the environment or *external verifier* (e.g., VAL (Howey et al., 2004)) to ensure they achieve the intended goal. When ground-truth plans are available, LLM-generated plans can also be *compared against these reference plans* (Zheng et al., 2024).

The second evaluation method is *human evaluation*, typically used in the following cases: (1) No available verifier: when certain simulated environments (e.g., VirtualHome) or real-world scenarios (e.g., using a mobile manipulator) lack automated verification; (2) Open-ended problems: tasks with ambiguous instructions or generative outputs (e.g., text or images) where multiple valid solutions may differ from the ground truth.

The final evaluation method, *LLM-as-a-Judge*, uses another LLM to automatically assess the quality of generated plans in the cases mentioned above. This approach has been increasingly adopted in recent LLM planning research (Guo et al., 2024; O'Donoghue et al., 2023). Compared to human

evaluation, LLM judges are faster and more cost-effective, making them especially valuable for evaluating large datasets. However, this method has limitations, such as position bias, length bias, self-inconsistency, and sensitivity to prompts (Zheng et al., 2023; Ye et al., 2024; Wei et al., 2024). Addressing these issues is crucial to ensure reliable assessments. For more details on LLM-as-a-Judge, please see Li et al. (2024a,b); Gu et al. (2024).

**Metrics** Figure 1 summarizes commonly used evaluation metrics for planning-focused tasks, along with representative works. Performance criteria are measured using specific metrics: (1) *Completeness*: success rate and goal condition recall measure whether the generated plan reaches final or stepwise goals, while classification metrics (e.g., true negative rate, false negative rate, and unreachable accuracy) assess the planner's ability to identify unachievable tasks. When ground-truth plans are available, the exact match score is used. (2) *Executability*: executability rate evaluates whether the plan can be executed in the environment, while constraint pass rate checks if constraints are met. (3) *Optimality*: measured by the optimality rate (i.e., the percentage of optimally solved tasks). (4) *Efficiency*: common metrics include inference time, input and output token counts, number of plan steps, and model size. (5) *Representation*: the number of parseable problems indicates correct translations. (6) *Generalization*: all these metrics can also be applied to unseen scenarios to assess generalization. See Figure 1 for definitions of individual metrics and representative works.

## 10 Discussion

In this section, we discuss the limitations in current LLM planning research studies, and suggest future directions for improvement and more comprehensive evaluations of LLM planning performance.

**Datasets and Baselines** The current evaluation of LLM planning has its limitations, primarily because studies often rely on limited datasets and baselines. This makes it tough to fairly and comprehensively compare different methods. Most studies only use a few datasets from a single domain and difficulty level, and they do not evaluate all the six performance criteria. Inconsistent dataset choices make direct comparisons difficult. On top of that, many studies only compare against basic baselines such as CoT or ReAct, which does not help in comparing more advanced approaches. To fix this, a public, standardized leaderboard should be set up that covers all performance criteria, uses consistent evaluation metrics, includes a variety of baseline and advanced methods, and utilizes diverse datasets spanning multiple domains and difficulty levels. Another useful direction would be to create multilingual planning datasets and assess LLM performance across different languages.

**Representation** LLMs are highly sensitive to prompts (Sclar et al., 2024; Razavi et al., 2025), but most research relies on natural language without comparing them to alternative formats, such as PDDL or Python, for describing domains and problems. Some studies (Singh et al., 2023; Aghzal et al., 2024) suggest that using Python to represent planning problems can improve performance, but automatically translating natural language problem descriptions into Python remains challenging, particularly for non-experts. If LLMs are to handle this translation effectively, additional datasets and evaluations are needed to assess their performance. Furthermore, little research has been conducted on how different prompt templates impact LLM planning performance, or on the best output formats for representing plans. Lastly, most fine-tuning methods rely on natural language data without exploring other formats, such as symbolic representations. Filling these gaps requires building benchmarks like Planetarium (Zuo et al., 2024) and carefully choosing representation formats in experiments.

**Hallucination** LLMs often experience hallucinations (Huang et al., 2023), which present two major challenges in planning. First, they might struggle to assess if a plan is achievable given a specific problem description (Aghzal et al., 2023; Kambhampati, 2024). Second, they can generate inadmissible actions and non-existent objects, requiring translation or expert intervention to correct them (Huang et al., 2022a; Raman et al., 2022). This increases the cost of planning systems. Further research is needed to understand the root causes and improve LLMs' ability to accurately identify unachievable plans. Evaluating the impact of these hallucinations remains an important research direction.

**Human Preference Alignment** There is a gap in understanding whether system generated plans align with human preferences. It is crucial for open-ended problems where humans execute the plans. Ensuring alignment with human preferences is vital for safety, feasibility, and usability, particularly in personalized planning tasks such as calendar and

travel planning. Additionally, Aghzal et al. (2024) found that LLM planners often fail to achieve optimality in path planning, frequently producing unnecessarily long plans. This may stem from inherent length bias in LLMs, which tend to generate longer sequences. Alignment techniques such as RLHF (Ouyang et al., 2022) and DPO (Rafailov et al., 2024) may help alleviate this issue, as humans generally prefer shorter plans for their efficiency, simplicity, and cognitive ease. Further investigation is needed to better align LLM planners with human preferences.

**Cost Effectiveness** Current methods, particularly task decomposition and search-based approaches, often consume a large number of tokens due to lengthy prompts and repeated LLM queries. While heuristic search is considered more efficient than task decomposition, it still requires substantial repeated prompting. To improve cost-effectiveness, we may summarize problem descriptions and enhance heuristic evaluations, e.g., by improving LLM uncertainty estimation (Huang et al., 2024a) and verification (Li et al., 2024c). These improvements would help reduce prompt length and enable the early stopping of unpromising partial plans.

**Multi-Agent Planning** Most existing research focuses on single-agent planning, where only one agent performs a task. Multi-agent planning (Konolige and Nilsson, 1980; Torreno et al., 2017) is more challenging, as it involves multiple agents (e.g., robots) working together or competing in parallel. Despite its complexity, multi-agent planning has received limited attention in AI planning research. It often requires coordinating multiple agents in collaborative or competitive environments where they operate simultaneously. The major challenge lies in developing effective communication protocols and cooperation strategies while generating viable plans for their collective actions.

**Reasoning, Tool Use, and Memory** There is often limited discussion on how other components of LLM agents, such as reasoning, tool use, and memory, affect planning performance. In particular, when LLMs are combined with classical planners or optimizers, it is crucial that the LLM accurately translates the planning problem into the appropriate domain representation to ensure correct plan generation. Currently, these approaches rely on human-selected planners and optimizers. Treating them as tools that LLMs can autonomously choose from could be an exciting prospect. This also raises the question of whether LLMs can effectively select the best tool for a given planning task. Future research should look into enhancing these agentic capabilities in LLM-based planning.

## 11 Conclusion

In our survey, we explore the landscape of modern LLM planners, proposing key performance criteria and discussing evaluation challenges. Our proposed criteria offer a structured approach to assess LLM planners across diverse domains. By systematically analyzing existing systems, datasets, and evaluation strategies, we aim to provide a foundation for future research in this space. We encourage researchers to build on our findings to create robust, highly adaptable, and efficient LLM planners.

## 12 Limitations

This work primarily focuses on commonly studied domains involving single-agent scenarios, such as robotics, household tasks, and computer-based tasks. We acknowledge that LLM planning is also applied in other areas, including the natural sciences (O'Donoghue et al., 2023; Liu et al., 2024), the Internet of Things (Cui et al., 2024), and multi-agent scenarios. However, these studies follow similar methodologies and evaluations, suggesting our survey's comprehensiveness. We focus on six commonly used performance criteria and exclude others, such as security and personalization, due to limited research in these areas. Instead, we discuss them in our future directions section.

## Acknowledgements

We thank the reviewers for their insightful feedback, which has helped improve our paper. H.W. and S.P. are supported by a UC Merced Spring 2023 Climate Action Seed Competition Grant, CAHSI-Google Institutional Research Program Award, and F3 R&D GSR Award funded by the US Department of Commerce, Economic Development Administration Build Back Better Regional Challenge. Z.Z. and F.L. are partially supported by NSF CAREER Award #2303655.

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2023. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning. *arXiv preprint arXiv:2310.03249*.

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2024. Look further ahead: Testing the limits of gpt-4 in path planning. *arXiv preprint arXiv:2406.12000*.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Vageesh D C, Arun Iyer, Suresh Parthasarathy, Sriram Rajamani, B. Ashok, and Shashank Shet. 2023. Codeplan: Repository-level coding using llms and planning. *Preprint*, arXiv:2309.12499.

Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, et al. 2021. The scip optimization suite 8.0. *arXiv preprint arXiv:2112.08872*.

Vineet Bhat, Ali Umut Kaypak, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. 2024. Grounding llms for robot task planning using closed-loop state feedback. *arXiv preprint arXiv:2402.08546*.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Faeze Brahman, Chandra Bhagavatula, Valentina Pyatkin, Jena D Hwang, Xiang Lorraine Li, Hirona Jacqueline Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. 2024. Plasma: Procedural knowledge models for language-based planning and re-planning. In *The Twelfth International Conference on Learning Representations*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Daniel Cao, Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi. 2024. Automating thought of search: A journey towards soundness and completeness. *arXiv preprint arXiv:2408.11326*.

Georgia Chalvatzaki, Ali Younes, Daljeet Nandha, An Thai Le, Leonardo FR Ribeiro, and Iryna Gurevych. 2023. Learning to reason over scene graphs: a case study of finetuning gpt-2 into a robot language model for grounded task planning. *Frontiers in Robotics and AI*, 10:1221739.

Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, et al. 2024. Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks. *arXiv preprint arXiv:2411.00081*.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023a. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

Jiangjie Chen, Siyu Yuan, Rong Ye, Bodhisattwa Prasad Majumder, and Kyle Richardson. 2023b. Put your money where your mouth is: Evaluating strategic planning and execution of llm agents in an auction arena. *arXiv preprint arXiv:2310.05746*.

Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2024a. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. In *2024 IEEE International conference on robotics and automation (ICRA)*, pages 6695–6702. IEEE.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024b. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*.

Zhenfang Chen, Delin Chen, Wenjun Sun, Rui abd Liu, and Chuang Gan. 2025. Autonomous agents from automatic reward modeling and planning. In *The Thirteenth International Conference on Learning Representations*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Hongwei Cui, Yuyang Du, Qun Yang, Yulin Shao, and Soung Chang Liew. 2024. Llmind: Orchestrating ai and iot with llm for complex task execution. *IEEE Communications Magazine*.

Aidan Curtis, Nishanth Kumar, Jing Cao, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2024. Trust the proc3s: Solving long-horizon robotics problems with llms and constraint satisfaction. *arXiv preprint arXiv:2406.05572*.

Gautier Dagan, Frank Keller, and Alex Lascarides. 2023. Dynamic planning with a llm. *arXiv preprint arXiv:2308.06391*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Alfonso Gerevini, Ivan Serina, et al. 2002. Lpg: A planner based on local search for planning graphs with action costs. In *Aips*, volume 2, pages 281–290.

Gonzalo Gonzalez-Pumariega, Wayne Chen, Kushal Kedia, and Sanjiban Choudhury. 2024. Query-efficient planning with language models. *arXiv preprint arXiv:2412.06162*.

Gonzalo Gonzalez-Pumariega, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. 2025. Robotouille: An asynchronous planning benchmark for llm agents. *arXiv preprint arXiv:2502.05227*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094.

Shiguang Guo, Ziliang Deng, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2024. Open grounded planning: Challenges and benchmark construction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 4982–5003. Association for Computational Linguistics.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. 2024a. Large language models can plan your travels rigorously with formal verification tools. *arXiv preprint arXiv:2404.11891*.

Yilun Hao, Yang Zhang, and Chuchu Fan. 2024b. Planning anything with rigor: General-purpose zero-shot planning with llm-based formalized programming. *arXiv preprint arXiv:2410.12112*.

Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. 2024. 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.

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

Eran Hirsch, Guy Uziel, and Ateret Anaby-Tavor. 2024. What's the plan? evaluating and developing planning-aware techniques for llms. *arXiv preprint arXiv:2402.11489*.

Richard Howey, Derek Long, and Maria Fox. 2004. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 294–301. IEEE.

Hanxu Hu, Hongyuan Lu, Huajian Zhang, Yun-Ze Song, Wai Lam, and Yue Zhang. 2023a. Chain-of-symbol prompting elicits planning in large langauge models. *arXiv preprint arXiv:2305.10276*.

Hanxu Hu, Hongyuan Lu, Huajian Zhang, Yun-Ze Song, Wai Lam, and Yue Zhang. 2024. Chain-of-symbol prompting for spatial reasoning in large language models. In *First Conference on Language Modeling*.

Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. 2023b. Tree-planner: Efficient close-loop task planning with large language models. *arXiv preprint arXiv:2310.08582*.

Hsiu-Yuan Huang, Yutong Yang, Zhaoxi Zhang, Sanwoo Lee, and Yunfang Wu. 2024a. A survey of uncertainty estimation in llms: Theory meets practice. *arXiv preprint arXiv:2410.15326*.

Jen-tse Huang, Eric John Li, Man Ho Lam, Tian Liang, Wenxuan Wang, Youliang Yuan, Wenxiang Jiao, Xing Wang, Zhaopeng Tu, and Michael R Lyu. 2024b. How far are we on the decision-making of llms? evaluating llms' gaming ability in multi-agent environments. *arXiv preprint arXiv:2403.11807*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

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

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

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024c. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Peter A Jansen. 2020. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. *arXiv preprint arXiv:2009.14259*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Da Ju, Song Jiang, Andrew Cohen, Aaron Foss, Sasha Mitts, Arman Zharmagambetov, Brandon Amos, Xian Li, Justine T Kao, Maryam Fazel-Zarandi, et al. 2024. To the globe (ttg): Towards language-driven guaranteed travel planning. *arXiv preprint arXiv:2410.16456*.

Subbarao Kambhampati. 2024. Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1):15–18.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*.

Nabil A Kartam and David E Wilkins. 1990. Towards a foundation for evaluating ai planners. *AI EDAM*, 4(1):1–13.

Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi. 2024. Thought of search: Planning with language models through the lens of efficiency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

Kurt Konolige and Nils J Nilsson. 1980. Multiple-agent planning systems. In *AAAI*, volume 80, pages 138–142.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. 2024. Beyond a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:2402.14083*.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2024a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024b. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

Haoming Li, Zhaoliang Chen, Songyuan Liu, Yiming Lu, and Fei Liu. 2024c. Systematic analysis of llm contributions to planning: Solver, verifier, heuristic. *Preprint*, arXiv:2412.09666.

Haoming Li, Zhaoliang Chen, Jonathan Zhang, and Fei Liu. 2024d. Lasp: Surveying the state-of-the-art in large language model-assisted ai planning. *arXiv preprint arXiv:2409.01806*.

Haoming Li, Zhaoliang Chen, Jonathan Zhang, and Fei Liu. 2025. Planet: A collection of benchmarks for evaluating llms' planning capabilities. *Preprint*, arXiv:2504.14773.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024e. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, et al. 2024f. Embodied agent interface: Benchmarking llms for embodied decision making. *arXiv preprint arXiv:2410.07166*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. 2022. Pretrained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212.

Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. 2023a. On grounded planning for embodied tasks with language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13192–13200.

Fangru Lin, Emanuele La Malfa, Valentin Hofmann, Elle Michelle Yang, Anthony Cohn, and Janet B Pierrehumbert. 2024. Graph-enhanced large language models in asynchronous plan reasoning. *arXiv preprint arXiv:2402.02805*.

Han Lin, Abhay Zala, Jaemin Cho, and Mohit Bansal. 2023b. Videodirectorgpt: Consistent multi-scene video generation via llm-guided planning. *arXiv preprint arXiv:2309.15091*.

Nir Lipovetzky, Miquel Ramirez, Christian Muise, and Hector Geffner. 2014. Width and inference based planners: Siw, bfs (f), and probe. *Proceedings of the 8th International Planning Competition (IPC-2014)*, page 43.

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

Gang Liu, Michael Sun, Wojciech Matusik, Meng Jiang, and Jie Chen. 2024. Multimodal large language models for inverse molecular design with retrosynthetic planning. *arXiv preprint arXiv:2410.04223*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Yujie Lu, Pan Lu, Zhiyu Chen, Wanrong Zhu, Xin Eric Wang, and William Yang Wang. 2023. Multimodal procedural planning via dual text-image prompting. *arXiv preprint arXiv:2305.01795*.

Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. 1998. Pddl-the planning domain definition language.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. *arXiv preprint arXiv:1904.03396*.

Allen Newell, John Calman Shaw, and Herbert A Simon. 1958. Elements of a theory of human problem solving. *Psychological review*, 65(3):151.

Odhran O'Donoghue, Aleksandar Shtedritski, John Ginger, Ralph Abboud, Ali Essa Ghareeb, Justin Booth, and Samuel G Rodriques. 2023. Bioplanner: automatic evaluation of llms on protocol planning in biology. *arXiv preprint arXiv:2310.10632*.

Jiefu Ou, Arda Uzunoglu, Benjamin Van Durme, and Daniel Khashabi. 2024. Worldapis: The world is worth how many apis? a thought experiment. *arXiv preprint arXiv:2407.07778*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jiayi Pan, Glen Chou, and Dmitry Berenson. 2023. Data-efficient learning of natural language to linear temporal logic translators for robot task specification. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11554–11561. IEEE.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2023. Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Raymond Mooney, Stefanie Tellex, and David Paulius. 2024. Cape: Corrective actions from precondition errors using large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14070–14077. IEEE.

Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. 2022. Planning with large language models via corrective re-prompting. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. 2025. Benchmarking prompt sensitivity in large language models. *Preprint*, arXiv:2502.06065.

Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. 2023. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Large language model-based ai agents for task planning and tool usage. *Preprint*, arXiv:2308.03427.

Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. pearson.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *Preprint*, arXiv:2310.11324.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. 2021. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*.

Zijing Shi, Meng Fang, and Ling Chen. 2025. Monte carlo planning with large language model for text-based games. In *The Thirteenth International Conference on Learning Representations*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020a. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020b. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Kaelbling, and Michael Katz. 2024. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20256–20264.

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

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*.

Siddharth Srivastava, Neil Immerman, and Shlomo Zilberstein. 2011. A new representation and associated algorithms for generalized planning. *Artificial Intelligence*, 175(2):615–647.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2024. Adaplanner: Adaptive planning from feedback with language models. *Advances in Neural Information Processing Systems*, 36.

Marcus Tantakoun, Xiaodan Zhu, and Christian Muise. 2025. Llms as planning modelers: A survey for leveraging large language models to construct automated planning models. *Preprint*, arXiv:2503.18971.

Alejandro Torreno, Eva Onaindia, Antonín Komenda, and Michal Štolba. 2017. Cooperative multi-agent planning: A survey. *ACM Computing Surveys (CSUR)*, 50(6):1–32.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005.

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

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *arXiv preprint arXiv:2402.11651*.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023b. Describe, explain, plan and select: Interactive planning with large language models enables open-worldzhao2024large multi-task agents. *arXiv preprint arXiv:2302.01560*.

Hui Wei, Shenghua He, Tian Xia, Andy Wong, Jingyang Lin, and Mei Han. 2024. Systematic evaluation of llm-as-a-judge in llm alignment tasks: Explainable metrics and diverse prompt templates. *arXiv preprint arXiv:2408.13006*.

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

Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, et al. 2024. Can graph learning improve planning in llm-based agents? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yue Wu, Xuan Tang, Tom M Mitchell, and Yuanzhi Li. 2023a. Smartplay: A benchmark for llms as intelligent agents. *arXiv preprint arXiv:2310.01557*.

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023b. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. 2023. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

Ruihan Yang, Jiangjie Chen, Yikai Zhang, Siyu Yuan, Aili Chen, Kyle Richardson, Yanghua Xiao, and Deqing Yang. 2024. Selfgoal: Your language agents already know how to achieve high-level goals. *arXiv preprint arXiv:2406.04784*.

Ryan Yang, Tom Silver, Aidan Curtis, Tomas Lozano-Perez, and Leslie Pack Kaelbling. 2022. Pg3: Policy-guided planning for generalized policy generation. *arXiv preprint arXiv:2204.10420*.

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

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, et al. 2024. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Quan Yuan, Mehran Kazemi, Xin Xu, Isaac Noble, Vaiva Imbrasaite, and Deepak Ramachandran. 2024. Tasklama: probing the complex task understanding of language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19468–19476.

Siyu Yuan, Jiangjie Chen, Ziquan Fu, Xuyang Ge, Soham Shah, Charles Robert Jankowski, Yanghua Xiao, and Deqing Yang. 2023. Distilling script knowledge from large language models for constrained language planning. *arXiv preprint arXiv:2305.05252*.

Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. 2023. Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. *arXiv preprint arXiv:2310.12128*.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, et al. 2024. Agentohana: Design unified data and training pipeline for effective agent learning. *arXiv preprint arXiv:2402.15506*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, et al. 2024. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023a. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023b. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2024. 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 2081–2088. IEEE.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. 2023. Toolchain*: Efficient action space navigation in large language models with a* search. *arXiv preprint arXiv:2310.13227*.

Max Zuo, Francisco Piedrahita Velez, Xiaochen Li, Michael L Littman, and Stephen H Bach. 2024. Planetarium: A rigorous benchmark for translating text to structured planning languages. *arXiv preprint arXiv:2407.03321*.

## 13 Appendix

Table 1: Summary of Foundations in LLM Planning (Section 2)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| DEPS (Wang et al., 2023b) | Alfworld | Success Rate, Average Episode Length | 1. LLM generates PDDL goal from task description<br>2. Sampling world beliefs and using symbolic planning<br>3. Plan generator (BFS(f)) for generating plans | 1.LLM-DP achieves 96% success in Alfworld, outperforming the ReAct baseline (53%) with fewer actions.<br>2.Enhances correctness in LLM planning by dynamically adjusting plans in response to environmental feedback and sub-goal feasibility. |
| ProgPrompt (Singh et al., 2023) | VirtualHome, Real-World Robot Tasks | Success Rate, Goal Conditions Recall, Executability | 1. Pythonic program generation for task planning<br>2. Use of natural language comments and assertions for feedback<br>3. Integration with real-time environment state feedback during task execution | 1. ProgPrompt significantly outperforms baseline methods by using programmatic LLM prompts to generate executable task plans, achieving up to 1.00 SR and 1.00 Exec in various VirtualHome tasks. It also adapts well to real-world robot tasks, with a Plan SR of 1 in most cases.<br>2. Improves correctness by incorporating environmental state feedback directly into the planning process. |
| Adaplanner (Sun et al., 2024) | ALFWorld and MiniWoB++ | Success Rate | Closed-loop approach allowing LLM agent to refine self-generated plan adaptively | 1. Uses code-style LLM prompt structure and skill discovery mechanism. Achieves 91.79% success rate on ALFWorld tasks.<br>2. Achieves 91.11% success rate on MiniWoB++ tasks with feedback.<br>3. Improves planning correctness by adaptively refining plans based on environmental feedback, effectively managing complex sequential tasks. |
| GNN-Enhanced Task Planner (Wu et al., 2024) | HuggingFace, Multimedia tasks, Daily Life API tasks, TMDB API tasks | Node F1-Score, Link F1-Score, Task Accuracy, Token Consumption | 1. Integration of GNNs for task graph navigation<br>2. Training-free (SGC) and training-required (GraphSAGE) approaches<br>3. GNN-based node and edge selection for task planning<br>4. Training fine-tuned models for better task retrieval | 1. GNN-enhanced planning outperforms LLM-based solutions by improving task accuracy (up to 9%) with reduced token consumption. The proposed method scales well with larger task graphs and improves planning efficiency by a significant margin.<br>2.Addresses correctness by effectively mapping task dependencies in a graph structure, enhancing sub-task selection and sequence planning. |
| SelfGoal (Yang et al., 2024) | Public Goods Game, Guess 2/3 of the Average, First-Price Auction, Bargaining | Success Rate, TrueSkill Score, Contribution Consistency | 1. Constructs GOALTREE to decompose high-level goals dynamically<br>2. Uses Search Module to select the most relevant subgoals<br>3. Decomposition updates based on environmental feedback<br>4. Adaptive subgoal tree refinement during task execution | 1. SelfGoal achieves a 94% success rate in dynamic multi-agent environments, outperforming baselines (e.g., ReAct, ADAPT) by dynamically adjusting subgoal guidance. It significantly improves task performance, especially in cooperative and competitive tasks.<br>2. SelfGoal enhances correctness and adaptability in planning by dynamically adjusting subgoals and actions based on ongoing task performance and feedback. |
| ADaPT (Prasad et al., 2023) | ALFWorld, WebShop, TextCraft & hotel data | Success Rate, Task Complexity Handling, Sub-Task Decomposition Efficiency | 1. As-needed recursive decomposition of complex tasks<br>2. Planner and executor modules with dynamic failure adaptation<br>3. Multi-level decomposition for task complexity handling | ADaPT improves success rates by up to 33% over baselines, dynamically decomposing complex tasks as needed. It handles task complexity efficiently with up to 28.3% higher success rates on ALFWorld and 27% higher on WebShop. |
| LLM+P (Liu et al., 2023a) | Blocksworld, Grippers, Barman, Termes, and other robot planning scenarios | Success Rate, Optimal Plan Success, Plan Execution Time | 1. Converts natural language problem description into PDDL format<br>2. Uses classical planners (e.g., FAST-DOWNWARD) to generate optimal plans<br>3. Translates planner output back into natural language | 1. LLM+P significantly outperforms LLMs in solving long-horizon planning problems, achieving optimal plans in robot domains with high success rates (up to 100% on Blocksworld) and minimal execution time. 2. LLM+P improves correctness by integrating LLMs with classical planning techniques to generate and execute optimal plans. |
| LLM-DP (Dagan et al., 2023) | Alfworld | Success Rate, Average Episode Length | 1. LLM converts task descriptions into executable PDDL goals<br>2. Symbolic planner (BFS(f)) generates valid plans<br>3. Belief sampling to generate multiple world states for planning | LLM-DP outperforms the ReAct baseline, achieving 96% success in Alfworld, with significantly fewer actions (13.16 vs. 18.69) per task and higher efficiency due to belief sampling and symbolic planning integration. |
| PDDL World Model Generation (Guan et al., 2023) | Household, Tyreworld, Logistics | Error Count, Success Rate | 1. LLM-based PDDL generation for task planning<br>2. Error correction using LLMs as feedback interfaces<br>3. Use of external planners for generating feasible plans from PDDL models | 1. The paper shows that GPT-4 is capable of generating high-quality PDDL models with fewer errors than GPT-3.5-Turbo. It demonstrates that GPT-4-based world models lead to a 95% success rate in planning tasks using classical planners like Fast Downward.<br>2. Enhances plan correctness by using LLM-generated PDDL models in conjunction with domain-independent planners, ensuring plans are not only feasible but also optimal. |
| LLM Planning Evaluation (Valmeekam et al., 2023b) | Blocksworld, Mystery Blocksworld | Success Rate, Levenshtein Edit Distance | 1. Evaluates LLMs (GPT-3, BLOOM) on common planning domains<br>2. Three evaluation modes: autonomous, heuristic, and human-in-the-loop<br>3. Comparison of LLM-generated plans with optimal solutions using automated planning tools | LLMs struggle with autonomous plan generation (3% success rate), but perform better when used for heuristic guidance in planning tasks (with LPG) and assist human planners by improving task completion accuracy (74% to 82%). |

Table 2: Summary of Foundations in LLM Planning (Section 2)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| Tree Search for Language Model Agents (Koh et al., 2024) | WebArena, VisualWebArena | Success Rate, Search Budget Efficiency | 1. Best-first search algorithm using language model feedback<br>2. Backtracking to refine paths based on value function<br>3. Model-based value function for search guidance | 1. Tree Search improves the GPT-4o agent's success rate by 39.7% on VisualWebArena and 28.0% on WebArena. The search mechanism enhances multi-step planning and exploration, setting a new state-of-the-art in web task completion.<br>2. Enhances correctness by using a best-first search strategy that adapts to environmental feedback, improving decision-making in complex web tasks. |
| SimPlan (Hirsch et al., 2024) | Blocksworld, Ferry, Grippers, Depots, Minigrid | Success Rate | 1. Hybrid approach combining LLMs with greedy best-first search<br>2. Utilizes external world modeling tools<br>3. Heuristic-driven planning for enhanced action ranking<br>4. Optimizes path selection through search algorithms | 1.SimPlan outperforms baseline LLM-based planners by integrating world modeling and search algorithms, achieving high success rates in domains like Blocksworld and Ferry. It highlights the effectiveness of combining LLMs with traditional planning methods.<br>2. Addresses limitations in LLMs' planning capabilities by integrating classical planning methodologies, enhancing correctness through structured problem-solving. |
| LATS (Zhou et al., 2023a) | HotPotQA, WebShop, HumanEval, Game of 24 | Pass@1 Accuracy, Success Rate | 1. Integrates MCTS-based search for reasoning, acting, and planning<br>2. External feedback integration with self-reflection for better decision-making<br>3. Dynamic sampling of multiple action paths for exploration | 1.LATS achieves state-of-the-art results, including 92.7% Pass@1 accuracy on HumanEval with GPT-4 and surpassing ReAct with 75.9% success rate on WebShop. It unifies reasoning, acting, and planning in one framework and improves decision-making by using Monte Carlo Tree Search.<br>2.Significantly enhances correctness in planning by introducing a tree search framework that adapts to environmental feedback and multiple reasoning paths. |
| Tree of Thought (Yao et al., 2024) | Game of 24, Creative Writing, Mini Crosswords | Success Rate, Passage Coherency, Task Completion Time | 1. Tree-based search for exploring multiple reasoning paths<br>2. Self-evaluation and decision-making with backtracking<br>3. Application of breadth-first and depth-first search<br>4. Thought decomposition for structured problem-solving | 1.ToT significantly improves problem-solving performance, with 74% success in Game of 24 compared to 4% for traditional Chain-of-Thought prompting. It introduces a new framework for deliberate planning and exploration in LLM-based reasoning tasks.<br>2.Enhances correctness by allowing for exploration of multiple paths and adjusting strategies based on real-time feedback. |
| Thought of Search (Katz et al., 2024) | 24 Game, Mini Crosswords, BlocksWorld, PrOntoQA | Success Rate, Plan Time, Model Evaluation Calls | 1. Using LLMs to generate code for search components (successor functions and goal tests)<br>2. Search performed using BFS and DFS<br>3. Minimizes LLM calls for efficiency | 1. Thought of Search achieves 100% accuracy across datasets with minimal LLM calls (1–2 calls per function). Significantly more efficient than existing LLM-based planning methods.<br>2. Significantly improves correctness by verifying search components for soundness and completeness, reducing computational inefficiency. |
| RAP (Hao et al., 2023) | Blocksworld, GSM8K, PrOntoQA | Success Rate, Plan Generation Success, Task Completion Time | 1. Repurposes LLM as both a world model and a reasoning agent<br>2. Uses Monte Carlo Tree Search (MCTS) for strategic exploration<br>3. Reward-based planning for balancing exploration and exploitation | 1. RAP achieves 33% relative improvement in plan generation over GPT-4 with CoT, and outperforms baseline methods on math reasoning and logical inference tasks, achieving a 64% success rate on Blocksworld and 94.2% accuracy on PrOntoQA.<br>2. Enhances correctness in LLM planning by effectively balancing exploration and exploitation in reasoning, leading to higher task success rates. |
| LLM-MCTS (Zhao et al., 2024) | Virtual Home | Success Rate, Task Completion Time, Task Complexity | 1. Combines LLM as a commonsense world model and heuristic policy in MCTS<br>2. Utilizes LLM to predict object locations and generate action suggestions<br>3. Improves planning efficiency by limiting search space with a heuristic policy<br>4. World model and policy guide MCTS to select promising action paths | LLM-MCTS achieves up to 91.4% success in simple tasks, outperforms GPT-3.5-based policies and traditional MCTS approaches, and demonstrates superior performance for large-scale task planning with partial observations. |
| Visual Semantic Planning with GPT-2 (Jansen, 2020) | ALFRED | Success Rate, Prediction Accuracy | 1. Uses GPT-2 for sequence-to-sequence translation of natural language directives into action plans<br>2. Evaluates command prediction accuracy using strict and permissive scoring metrics<br>3. Focuses on generating multi-step plans with minimal visual input | 1. The GPT-2 model achieved 26% accuracy in generating correct visual semantic plans without visual input, increasing to 58% when provided with starting location. This demonstrates that detailed task planning can be done using language models alone.<br>2. Improves correctness and efficiency in task planning by leveraging LLMs for both heuristic guidance and world modeling, leading to better performance in complex tasks. |
| RobLM (Chalvatzaki et al., 2023) | ALFRED | Task Success Rate, Plan Accuracy, Argument Accuracy, Sub-Task Success Rate | 1. Fine-tunes GPT-2 for task planning with grounded scene graph input<br>2. Uses Graph2NL to convert scene graphs into natural language for model input<br>3. Task and geometric grounding for robot execution<br>4. Grounding high-level actions to low-level robotic commands | 1.RobLM outperforms classical planning systems in several task categories, achieving high success rates (up to 98% for "SliceObject") and demonstrating the feasibility of grounded LLMs for robotic task planning.<br>2. Demonstrates correctness in generating actionable plans from textual directives alone, significantly improving plan success rates without relying on visual data. |

Table 3: Summary of Foundations in LLM Planning (Section 2)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| Agent-FLAN (Chen et al., 2024b) | HALF-World, WebShop, Mind2Web, Knowledge Graph, Operating System, Database, ToolBench | Accuracy, hallucination scores | 1.Fine-tuning LLMs (Llama2-7B) for agent tasks<br>2.Decomposing training data into capabilities like reasoning, retrieval, understanding, instruction following<br>3. Negative sample generation for hallucination mitigation | 1. Outperforms previous works by 3.5% on agent tasks<br>2. Significantly reduces hallucinations through negative sample learning<br>3. Demonstrates scaling law for data and model size improvement<br>4. Enhances general capabilities like reasoning and instruction-following in general NLP tasks |
| AgentOhana (Zhang et al., 2024) | Webshop, HotpotQA, ToolAlpaca, ToolBench, Alf-World, APIbank, Mind2Web, Knowledge Graph, Database | Success Rate, Task Completion Time, Reward Accuracy | 1. Standardizes heterogeneous multi-turn agent trajectories from diverse environments<br>2. AgentRater for filtering low-quality trajectories<br>3. Generic dataloader for efficient data handling during training | AgentOhana aggregates and unifies agent trajectories from ten diverse environments, achieving significant improvements in model performance and task execution. The xLAM-v0.1 model trained using AgentOhana outperforms GPT-4 and other models in benchmarks like Webshop and ToolEval. |
| NAT (Wang et al., 2024) | GSM8K, ASDiv, SVAMP, MultiArith, HotpotQA, StrategyQA | Accuracy, F1-Score, EM Score | 1. Incorporates both positive and negative samples in fine-tuning LLMs for agent tasks<br>2. Introduces a prefix/suffix to indicate whether trajectories are positive or negative<br>3. Differentiates high-quality and low-quality negative examples<br>4. Fine-grained NAT with two-level classification of negative examples | 1. NAT achieves a 64.64% average performance across GSM8K, ASDiv, SVAMP, and MultiArith using 2k positive and 10k negative samples. It outperforms Vanilla and NUT methods in various tasks, demonstrating the value of negative examples for agent fine-tuning.<br>2.Enhances correctness by utilizing negative examples to inform the fine-tuning process, improving task success rates |
| ETO (Song et al., 2024) | WebShop, ScienceWorld, ALFWorld | Average Reward, Success Rate, Task Solving Efficiency | 1. Exploration phase for failure trajectory collection<br>2. Contrastive learning using failure-success trajectory pairs<br>3. DPO loss for iterative policy refinement<br>4. Iterative optimization cycle for improvement | 1. ETO demonstrates a 22% improvement in task-solving efficiency over traditional behavioral cloning, surpassing baselines like PPO and RFT. It significantly improves performance in both in-domain and out-of-domain tasks, especially when expert trajectories are unavailable.<br>2. Significantly improves correctness in task planning by allowing agents to learn from both successful and failed interactions, leading to robust policy enhancements. |
| MC-DML (Shi et al., 2025) | Jericho Benchmark | Success Rate, Planning Efficiency | 1. Integrates Monte Carlo Tree Search (MCTS) with LLMs for more effective planning.<br>2. Uses in-trial and cross-trial memory mechanisms to enhance decision-making based on past failures. | 1. Strengthens the foundation of LLM planning by combining search algorithms (MCTS) with language-based reasoning, enabling more structured and adaptive decision-making.<br>2. Improves correctness by leveraging memory-guided LLM policies to adjust action probabilities dynamically, ensuring valid and effective decision paths in text-based planning. |
| ARMAP (Chen et al., 2025). | Multiple agent benchmarks | Success Rate | 1. Uses LLMs to navigate environments and generate diverse action trajectories.<br>2. Employs a separate LLM to create and refine a reward model from these trajectories, integrating it with various planning algorithms. | 1. Establishes a novel foundation for LLM planning by automating the creation of reward models, enhancing decision-making capabilities of LLM agents.<br>2. Demonstrates a significant advancement in LLM agents' ability to handle complex, multi-step decision-making tasks without human-annotated data, making the approach highly scalable and adaptable. |

Table 4: Summary of Completeness in LLM Planning (Section 3)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| PDDL World Model Generation (Guan et al., 2023) | Household, Tyreworld, Logistics | Error Count, Success Rate | 1. LLM-based PDDL generation for task planning<br>2. Error correction using LLMs as feedback interfaces<br>3. Use of external planners for generating feasible plans from PDDL models | The paper shows that GPT-4 is capable of generating high-quality PDDL models with fewer errors than GPT-3.5-Turbo. It demonstrates that GPT-4-based world models lead to a 95% success rate in planning tasks using classical planners like Fast Downward. |
| Large Language Models Can Solve Real-World Planning Rigorously with Formal Verification Tools (Hao et al., 2024a) | TravelPlanner, UnsatChristmas, TSP, Block Picking, Task Allocation, Warehouse | Success Rate, Optimization Rate, Interactive Repair Success Rate | 1. Converts natural language travel planning queries into SMT (Satisfiability Modulo Theory) problems<br>2. Uses LLMs to generate steps and code for formal solvers<br>3. Provides feedback and suggestions for unsatisfiable queries<br>4. Zero-shot generalization to new tasks and constraints | 1.TravelPlan-LLM achieves 93.9% success in solving complex multi-constraint travel planning problems and offers interactive plan repair for unsatisfiable queries. It generalizes well to new problem domains, including TSP and task allocation.<br>2.Enhances completeness by using formal tools to verify that LLM-generated plans are both correct and complete, ensuring that all solutions are viable and no false solutions are proposed. |
| PPNL (Aghzal et al., 2023) | Grid environments, ALFRED | Success Rate, Optimal Rate, Exact Match Accuracy, Unreachable Accuracy, Feasible Rate | 1. Few-shot prompting with GPT-4 for path planning<br>2. Action-and-effect prompting to guide long-term spatial reasoning<br>3. CoT and ReAct prompting to enhance spatial-temporal reasoning and obstacle avoidance<br>4. Multi-goal path planning with hierarchical task decomposition | 1.PPNL evaluates LLMs on path planning tasks, demonstrating GPT-4's spatial reasoning ability when prompted effectively. ReAct prompting achieves 96.1% success rate, while fine-tuned models like T5 outperform in in-distribution settings but struggle with generalization.<br>2. Directly addresses the completeness of LLM planning by evaluating their ability to identify unsolvable problems and correctly navigate solvable scenarios. |
| LRM Evaluation (Valmeekam et al., 2024) | Blocksworld, Mystery Blocksworld, Randomized Mystery Blocksworld | Success Rate, Execution Time | 1. Comparison of LLMs and LRMs using the PlanBench test set<br>2. Zero-shot and one-shot prompting for evaluation<br>3. Performance measured on standard Blocksworld tasks as well as obfuscated versions (Mystery Blocksworld)<br>4. Efficiency evaluation of reasoning tokens used by o1 | LRM o1 outperforms all LLMs, achieving 97.8% on Blocksworld, but struggles with performance on Mystery Blocksworld (52.8d%). LLMs fail to generalize well, showing significant limitations when faced with obfuscated tasks. |

Table 5: Summary of Executability in LLM Planning (Section 4)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| Adaplanner (Sun et al., 2024) | ALFWorld and MiniWoB++ | Success Rate | Closed-loop approach allowing LLM agent to refine self-generated plan adaptively | • Uses code-style LLM prompt structure and skill discovery mechanism. Achieves 91.79% success rate on ALFWorld tasks.<br>• Achieves 91.11% success rate on MiniWoB++ tasks with feedback. |
| Inner Monologue (Huang et al., 2022b) | Tabletop and mobile manipulation tasks in both simulated and real-world settings | Success Rate | Uses LLMs for generating sequential, actionable plans based on feedback.<br>Employs closed-loop feedback incorporating success detection, scene descriptions, and human interactions. | Demonstrates that LLMs can effectively integrate multimodal feedback to improve planning and execution in robotic tasks.<br>Shows adaptability of LLMs to dynamic environments and tasks, significantly enhancing task success rates in complex manipulation scenarios. |
| LLM-Planner (Song et al., 2023) | ALFRED | Success Rate | Uses GPT-3 for generating high-level plans.<br>Incorporates dynamic grounded re-planning based on environment feedback. | Achieves high few-shot performance on the ALFRED dataset, demonstrating the ability to quickly adapt to new tasks with minimal training data.<br>Introduces dynamic re-planning to adjust plans based on real-time environmental changes, enhancing task success under varied conditions. |
| G-PlanET (Lin et al., 2023a) | ALFRED | Key Action Score | Utilizes encoder-decoder LMs with a focus on grounded planning for embodied tasks.<br>Introduces object tables as environmental input for LMs to perceive and plan actions. | First study to investigate LMs' capability in grounded planning for embodied tasks.<br>Developed a new evaluation metric (KAS) tailored for assessing the quality of generated plans in realistic environments. |
| LLP (Sharma et al., 2021) | ALFRED | Task Completion rates | Utilizes latent language to segment and label hierarchical tasks in demonstrations.<br>Employs weak and partial natural language supervision to train policies. | Demonstrates effective policy training with minimal natural language annotations, achieving performance comparable to models using more extensive data.<br>Introduces a method that allows hierarchical policies to be trained using unannotated action sequences by inferring natural language descriptions. |
| SayCan (Ahn et al., 2022) | Real-World Kitchen environment | Plan Success Rate, Execution Success Rate | Leverages LLMs to score the likelihood of robotic skills contributing to task completion.<br>Combines LLM outputs with affordance functions from reinforcement learning to prioritize feasible actions. | Enables robots to execute complex, long-horizon tasks using natural language instructions.<br>Nearly doubles performance over non-grounded baselines by integrating real-world affordance functions with LLM predictions. |
| ADaPT (Prasad et al., 2023) | ALFWorld, WebShop, TextCraft & hotel data | Success Rate, Task Complexity Handling, Sub-Task Decomposition Efficiency | 1. As-needed recursive decomposition of complex tasks<br>2. Planner and executor modules with dynamic failure adaptation<br>3. Multi-level decomposition for task complexity handling | ADaPT improves success rates by up to 33% over baselines, dynamically decomposing complex tasks as needed. It handles task complexity efficiently with up to 28.3% higher success rates on ALFWorld and 27% higher on WebShop. |
| LLM-DP (Dagan et al., 2023) | Alfworld | Success Rate, Average Episode Length | 1. LLM converts task descriptions into executable PDDL goals<br>2. Symbolic planner (BFS(f)) generates valid plans<br>3. Belief sampling to generate multiple world states for planning | LLM-DP outperforms the ReAct baseline, achieving 96% success in Alfworld, with significantly fewer actions (13.16 vs. 18.69) per task and higher efficiency due to belief sampling and symbolic planning integration. |
| LLM-MCTS (Zhao et al., 2024) | Virtual Home | Success Rate, Task Completion Time, Task Complexity | 1. Combines LLM as a commonsense world model and heuristic policy in MCTS<br>2. Utilizes LLM to predict object locations and generate action suggestions<br>3. Improves planning efficiency by limiting search space with a heuristic policy<br>4. World model and policy guide MCTS to select promising action paths | LLM-MCTS achieves up to 91.4% success in simple tasks, outperforms GPT-3.5-based policies and traditional MCTS approaches, and demonstrates superior performance for large-scale task planning with partial observations. |
| Corrective Re-prompting (Raman et al., 2022) | Virtual Home | Executability Rate, Semantic Correctness, Number of Re-prompts | 1. Utilizes precondition errors for re-prompting to generate corrective actions.<br>Employs a template-based prompt strategy that incorporates error details for plan adjustments. | 1. Introduced a novel approach to improve plan executability and correctness using error-driven re-prompting.<br>2. Demonstrated substantial improvements in task execution rates and reduced the number of necessary re-prompts compared to baselines. |

Table 6: Summary of Executability in LLM Planning (Section 4)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| LLM-Planner (Song et al., 2023) | ALFRED | Success Rate | Uses GPT-3 for generating high-level plans. Incorporates dynamic grounded re-planning based on environment feedback. | Achieves high few-shot performance on the ALFRED dataset, demonstrating the ability to quickly adapt to new tasks with minimal training data. Introduces dynamic re-planning to adjust plans based on real-time environmental changes, enhancing task success under varied conditions. |
| SayCanPay (Hazra et al., 2024) | Ravens (Tower of Hanoi), BabyAI, VirtualHome | Planning Success, Cost-Effectiveness, Generalization | 1. Heuristic search-based planning framework using LLMs 2. Three-step process: Say (generate actions), Can (action feasibility), Pay (action payoff) 3. Beam search for action selection | SayCanPay outperforms other LLM-based planning approaches, achieving higher planning success rates (e.g., 94% on BabyAI), improved cost-efficiency, and better generalization across environments. |
| BrainBody-LLM (Bhat et al., 2024) | Virtual Home, Franka Research 3 | Success Rate, Goal Condition Recall | Employs two separate LLMs for high-level planning and low-level control. Utilizes closed-loop state feedback to iteratively refine plans based on environmental feedback. | - Achieved a 29% improvement in task-oriented success rates over existing methods. Demonstrated effective grounding of language models in robotic task planning with minimal human intervention. |
| CLP (Yuan et al., 2023) | CoScript 3 | Constraint faithfulness, Executability rate | 1.Uses an over-generate-then-filter approach to improve LLMs' planning. 2. Distills structured script knowledge into smaller models using LLMs. | 1.Introduced constrained language planning as a new task, allowing for step-by-step procedural reasoning under constraints. 2.Proposed CoScript, a dataset of 55,000 constraint-driven scripts, improving planning performance. 3.Smaller models trained on CoScript outperform LLMs in constrained planning accuracy. |
| PLaSma (Brahman et al., 2024) | COPLAN, VirtualHome | Human Evaluation: Coverage, Order, Overall Quality, Executability, Correctness | 1. Symbolic procedural knowledge distillation 2. Multi-task and task-specific distillation 3. Verifier-guided step-wise beam search 4. Constrained and counterfactual replanning tasks | PlaSma enhances smaller models with procedural knowledge, outperforming GPT-3 in tasks like goal-based planning and counterfactual replanning. Achieves up to 93% success in counterfactual replanning. |
| PRoC3S (Curtis et al., 2024) | Various simulated and real-world domains 3 | Success Rate | 1. Uses LLMs to generate programs handling continuous parameters and constraints. 2.Incorporates continuous constraint satisfaction to adjust plans based on real-time feedback. | 1. Pioneered the integration of LLMs with constraint satisfaction techniques for complex robotic tasks. 2.Demonstrated superior efficiency and effectiveness in diverse manipulation tasks compared to existing methods. |
| BrainBody-LLM (Bhat et al., 2024) | Virtual Home, Franka Research 3 | Success Rate, Goal Condition Recall | Employs two separate LLMs for high-level planning and low-level control. Utilizes closed-loop state feedback to iteratively refine plans based on environmental feedback. | - Achieved a 29% improvement in task-oriented success rates over existing methods. Demonstrated effective grounding of language models in robotic task planning with minimal human intervention. |
| ProgPrompt (Singh et al., 2023) | VirtualHome, Real-World Robot Tasks | Success Rate, Goal Conditions Recall, Executability | 1. Pythonic program generation for task planning 2. Use of natural language comments and assertions for feedback 3. Integration with real-time environment state feedback during task execution | ProgPrompt significantly outperforms baseline methods by using programmatic LLM prompts to generate executable task plans, achieving up to 1.00 SR and 1.00 Exec in various VirtualHome tasks. It also adapts well to real-world robot tasks, with a Plan SR of 1 in most cases. |
| ISR-LLM (Zhou et al., 2024) | Diverse planning problem domains | Success Rate | 1.Utilizes iterative self-refinement to enhance LLM-based planning. 2.Employs an LLM translator to convert natural language to PDDL, aiding in plan formulation and refinement. | 1.Introduced a novel framework that significantly improves feasibility and success in long-horizon task planning. 2. Demonstrated superior task accomplishment rates across multiple domains compared to state-of-the-art LLM-based planners. |
| ADaPT (Prasad et al., 2023) | ALFWorld, WebShop, TextCraft & hotel data | Success Rate, Task Complexity Handling, Sub-Task Decomposition Efficiency | 1. As-needed recursive decomposition of complex tasks 2. Planner and executor modules with dynamic failure adaptation 3. Multi-level decomposition for task complexity handling | ADaPT improves success rates by up to 33% over baselines, dynamically decomposing complex tasks as needed. It handles task complexity efficiently with up to 28.3% higher success rates on ALFWorld and 27% higher on WebShop. |
| SelfGoal (Yang et al., 2024) | Public Goods Game, Guess 2/3 of the Average, First-Price Auction, Bargaining | Success Rate, TrueSkill Score, Contribution Consistency | 1. Constructs GOALTREE to decompose high-level goals dynamically 2. Uses Search Module to select the most relevant subgoals 3. Decomposition updates based on environmental feedback 4. Adaptive subgoal tree refinement during task execution | SelfGoal achieves a 94% success rate in dynamic multi-agent environments, outperforming baselines (e.g., ReAct, ADAPT) by dynamically adjusting subgoal guidance. It significantly improves task performance, especially in cooperative and competitive tasks. |

Table 7: Summary of Optimality in LLM Planning (Section 5)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| TTG (Ju et al., 2024) | Synthetic travel requests and flight & hotel data | Exact Match accuracy, Cost ratio, Net Promoter Score | 1. Fine-tuned LLM for translating NL requests to symbolic form<br>2. MILP solver for optimal travel itineraries<br>3. Real-time response (<5 seconds) with guaranteed optimality | TTG achieves 91% translation accuracy with LLM, guarantees optimal itineraries with minimal cost, and provides a user-friendly system with a high NPS (35-40%) on generated itineraries. |
| ToolChain* (Zhuang et al., 2023) | Home Search, Trip Booking, Google Sheets, Virtual Home, GSM8K | Success Rate, Efficiency (Time), Running Time Comparison | 1. A* search-based planning algorithm for tool-use tasks<br>2. Efficient node expansion and pruning using task-specific cost functions<br>3. Combines exploration and exploitation for optimal solutions | ToolChain* improves success rate by 3.1% and reduces planning time by 7.35x compared to baselines like MCTS, demonstrating its efficiency in navigating expansive action spaces. |
| SayCanPay (Hazra et al., 2024) | Ravens (Tower of Hanoi), BabyAI, VirtualHome | Planning Success, Cost-Effectiveness, Generalization | 1. Heuristic search-based planning framework using LLMs<br>2. Three-step process: Say (generate actions), Can (action feasibility), Pay (action payoff)<br>3. Beam search for action selection | SayCanPay outperforms other LLM-based planning approaches, achieving higher planning success rates (e.g., 94% on BabyAI), improved cost-efficiency, and better generalization across environments. |
| Beyond A* (Lehnert et al., 2024) | Sokoban, Maze Navigation | Success Weighted by Cost (SWC), Improved Length Ratio (ILR), Optimal Plan Success Rate | 1. Trains Transformer to imitate A* search dynamics<br>2. Fine-tunes to generate shorter execution traces<br>3. Bootstraps from search dynamics for optimized plan generation | 1. Demonstrates novel training methods that integrate search dynamics into Transformer training, enhancing planning efficiency.<br>2. Achieves significant reductions in search steps (up to 26.8% fewer than A*) and improves task-solving performance on complex puzzles and navigation tasks. |
| LLMFP (Hao et al., 2024b) | 9 diverse planning tasks (multi-constraint decision-making and multi-step planning) | Plan optimality, constraint satisfaction rate | 1. Converts natural language planning problems into formal optimization problems using zero-shot LLM reasoning.<br>2. Uses SMT solvers to guarantee that the generated plans are both logically correct and executable. | 1. Introduces a novel method of achieving optimality in LLM planning by rigorously constructing optimization problems from natural language inputs, significantly outperforming traditional planning models.<br>2. Demonstrates strong performance across a range of complex tasks, achieving over 83% optimal rates, thereby highlighting the effectiveness of integrating LLMs with classical optimization approaches for high-quality planning outcomes. |

Table 8: Summary of Representation in LLM Planning (Section 6)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| LLM-GoalTrans (Xie et al., 2023) | Blocksworld, ALFRED | Goal Translation Success Rate | 1. Uses GPT-3.5 to translate natural language instructions into PDDL goals.<br>2.Employs n-shot learning to improve translation accuracy. | 1. Demonstrated effective translation of natural language to PDDL goals, with high success rates in structured environments.<br>2.Highlighted the challenges and limitations of LLMs in tasks requiring numerical or physical reasoning. |
| ISR-LLM (Zhou et al., 2024) | Diverse planning problem domains | Success Rate | 1.Utilizes iterative self-refinement to enhance LLM-based planning.<br>2.Employs an LLM translator to convert natural language to PDDL, aiding in plan formulation and refinement. | 1.Introduced a novel framework that significantly improves feasibility and success in long-horizon task planning.<br>2. Demonstrated superior task accomplishment rates across multiple domains compared to state-of-the-art LLM-based planners. |
| Adaplanner (Sun et al., 2024) | ALFWorld and MiniWoB++ | Success Rate | Closed-loop approach allowing LLM agent to refine self-generated plan adaptively | 1. Uses code-style LLM prompt structure and skill discovery mechanism. Achieves 91.79% success rate on ALFWorld tasks.<br>2. Achieves 91.11% success rate on MiniWoB++ tasks with feedback. |
| LLM-GenPlan (Silver et al., 2024) | Various PDDL domains | Percentage of tasks solved | 1.Uses GPT-4 to synthesize domain-specific Python programs for task planning.<br>2.Incorporates Chain-of-Thought summarization and automated debugging in the process. | 1.Demonstrated GPT-4's effectiveness as a generalized planner across several PDDL domains.<br>2.Highlighted the significant benefits of automated debugging and the mixed impacts of Chain-of-Thought summarization in planning tasks. |
| LLM+P (Liu et al., 2023a) | Blocksworld, Grippers, Barman, Termes, and other robot planning scenarios | Success Rate, Optimal Plan Success, Plan Execution Time | 1. Converts natural language problem description into PDDL format<br>2. Uses classical planners (e.g., FAST-DOWNWARD) to generate optimal plans<br>3. Translates planner output back into natural language | LLM+P significantly outperforms LLMs in solving long-horizon planning problems, achieving optimal plans in robot domains with high success rates (up to 100% on Blocksworld) and minimal execution time. |
| PDDL World Model Generation (Guan et al., 2023) | Household, Tyreworld, Logistics | Error Count, Success Rate | 1. LLM-based PDDL generation for task planning<br>2. Error correction using LLMs as feedback interfaces<br>3. Use of external planners for generating feasible plans from PDDL models | The paper shows that GPT-4 is capable of generating high-quality PDDL models with fewer errors than GPT-3.5-Turbo. It demonstrates that GPT-4-based world models lead to a 95% success rate in planning tasks using classical planners like Fast Downward. |
| LLM-DP (Dagan et al., 2023) | Alfworld | Success Rate, Average Episode Length | 1. LLM converts task descriptions into executable PDDL goals<br>2. Symbolic planner (BFS(f)) generates valid plans<br>3. Belief sampling to generate multiple world states for planning | LLM-DP outperforms the ReAct baseline, achieving 96% success in Alfworld, with significantly fewer actions (13.16 vs. 18.69) per task and higher efficiency due to belief sampling and symbolic planning integration. |
| LTL-Gen (Pan et al., 2023) | Custom datasets with LTL/natural language pairs | Accuracy | 1. Uses large-scale semantic parsing with minimal human-labeled data.<br>2.Employs synthetic data generation and constrained decoding. | 1.Demonstrates high accuracy with minimal human data compared to prior work.<br>2.Enhances data efficiency for natural language to LTL translation, making it feasible for broader applications. |
| AutoTAMP (Chen et al., 2024a) | Custom 2D task domains | Success Rate | 1. Translates NL instructions to STL using LLMs.<br>2.Utilizes autoregressive re-prompting for syntactic and semantic error correction.<br>3.Plans trajectories with a formal STL planner. | 1. Introduces a novel method for autoregressive re-prompting to correct translation errors.<br>2.Demonstrates significant improvements in task success rates with hard geometric and temporal constraints.<br>3.Provides a robust framework that combines LLM translation capabilities with formal planning efficiency. |
| G-PlanET (Lin et al., 2023a) | ALFRED | Key Action Score | Utilizes encoder-decoder LMs with a focus on grounded planning for embodied tasks.<br>Introduces object tables as environmental input for LMs to perceive and plan actions. | First study to investigate LMs' capability in grounded planning for embodied tasks.<br>Developed a new evaluation metric (KAS) tailored for assessing the quality of generated plans in realistic environments. |
| PPNL (Aghzal et al., 2023) | Grid environments, ALFRED | Success Rate, Optimal Rate, Exact Match Accuracy, Unreachable Accuracy, Feasible Rate | 1. Few-shot prompting with GPT-4 for path planning<br>2. Action-and-effect prompting to guide long-term spatial reasoning<br>3. CoT and ReAct prompting to enhance spatial-temporal reasoning and obstacle avoidance<br>4. Multi-goal path planning with hierarchical task decomposition | PPNL evaluates LLMs on path planning tasks, demonstrating GPT-4's spatial reasoning ability when prompted effectively. ReAct prompting achieves 96.1% success rate, while fine-tuned models like T5 outperform in in-distribution settings but struggle with generalization. |
| ProgPrompt (Singh et al., 2023) | VirtualHome, Real-World Robot Tasks | Success Rate, Goal Conditions Recall, Executability | 1. Pythonic program generation for task planning<br>2. Use of natural language comments and assertions for feedback<br>3. Integration with real-time environment state feedback during task execution | ProgPrompt significantly outperforms baseline methods by using programmatic LLM prompts to generate executable task plans, achieving up to 1.00 SR and 1.00 Exec in various VirtualHome tasks. It also adapts well to real-world robot tasks, with a Plan SR of 1 in most cases. |
| Adaplanner (Sun et al., 2024) | ALFWorld and MiniWoB++ | Success Rate | Closed-loop approach allowing LLM agent to refine self-generated plan adaptively | 1. Uses code-style LLM prompt structure and skill discovery mechanism. Achieves 91.79% success rate on ALFWorld tasks.<br>2. Achieves 91.11% success rate on MiniWoB++ tasks with feedback. |
| LID (Li et al., 2022) | VirtualHome, BabyAI | Task completion rates, Generalization to novel scenes and tasks | 1. Utilizes pre-trained LMs to convert observations, goals, and history into sequential data for decision-making.<br>2. Employs active data gathering to train policies without pre-collected expert data. | 1. Demonstrates that pre-trained LMs can significantly improve task completion rates and generalization in interactive decision-making.<br>2. Provides insights into the effectiveness of sequential input representations and LM-based weight initialization for generalization. |
| SayCan (Ahn et al., 2022) | Real-World Kitchen environment | Plan Success Rate, Execution Success Rate | Leverages LLMs to score the likelihood of robotic skills contributing to task completion.<br>Combines LLM outputs with affordance functions from reinforcement learning to prioritize feasible actions. | Enables robots to execute complex, long-horizon tasks using natural language instructions.<br>Nearly doubles performance over non-grounded baselines by integrating real-world affordance functions with LLM predictions. |

Table 9: Summary of Representation in LLM Planning (Section 6)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| PLaG (Lin et al., 2024) | AsyncHow | accuracy | 1. Utilizes graph-based representations to instruct LLMs in planning tasks.<br>2. Combines natural language with graph theory for enhanced task planning. | 1. Successfully introduces a graph-theoretical approach to enhance LLMs' performance in asynchronous planning tasks.<br>2. Demonstrates the utility of graph-enhanced prompts to improve accuracy across different LLM architectures. |
| GNN-Enhanced Task Planner (Wu et al., 2024) | HuggingFace, Multimedia tasks, Daily Life API tasks, TMDB API tasks | Node F1-Score, Link F1-Score, Task Accuracy, Token Consumption | 1. Integration of GNNs for task graph navigation<br>2. Training-free (SGC) and training-required (GraphSAGE) approaches<br>3. GNN-based node and edge selection for task planning<br>4. Training fine-tuned models for better task retrieval | GNN-enhanced planning outperforms LLM-based solutions by improving task accuracy (up to 9%) with reduced token consumption. The proposed method scales well with larger task graphs and improves planning efficiency by a significant margin. |

Table 10: Summary of Generalization in LLM Planning (Section 7)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| LLM-GenPlan (Silver et al., 2024) | Various PDDL domains | Percentage of tasks solved | 1.Uses GPT-4 to synthesize domain-specific Python programs for task planning.<br>2.Incorporates Chain-of-Thought summarization and automated debugging in the process. | 1.Demonstrated GPT-4's effectiveness as a generalized planner across several PDDL domains.<br>2.Highlighted the significant benefits of automated debugging and the mixed impacts of Chain-of-Thought summarization in planning tasks. |
| VOYAGER (Wang et al., 2023a) | Minecraft | Unique items collected, Distance traveled, Tech tree milestones | 1. Uses an automatic curriculum to maximize exploration.<br>2. Maintains a skill library for storing and retrieving executable behaviors.<br>3. Employs an iterative prompting mechanism that incorporates feedback for continuous improvement. | 1. First LLM-powered embodied agent for lifelong learning in Minecraft, capable of continuous self-improvement.<br>2. Demonstrates significant improvements in discovery and task completion, outperforming state-of-the-art methods in efficiency and generalization. |

Table 11: Summary of Efficiency in LLM Planning (Section 8)

| Method Name | Dataset | Evaluation Metric | Methods | Major Contribution |
|---|---|---|---|---|
| Adaplanner (Sun et al., 2024) | ALFWorld and MiniWoB++ | Success Rate | Closed-loop approach allowing LLM agent to refine self-generated plan adaptively | 1. Uses code-style LLM prompt structure and skill discovery mechanism. Achieves 91.79% success rate on ALFWorld tasks.<br>2. Achieves 91.11% success rate on MiniWoB++ tasks with feedback.<br>3. Improves planning correctness by adaptively refining plans based on environmental feedback, effectively managing complex sequential tasks. |
| Query-Efficient Planning (Gonzalez-Pumariega et al., 2024) | PlanBench, Logistics, Grippers, Robotouille | Success Rate | 1. Uses LLMs as heuristics in search-based planning to propose actions or select states.<br>2. Employs generative LLM planners that dynamically adapt plans based on feedback from a world model. | 1. Enhances query efficiency in planning by leveraging LLMs to reduce the number of queries to the world model, thus saving computational resources and time.<br>2. Demonstrates that LLM-based generative planners can effectively adapt to feedback, improving correctness and efficiency in planning tasks by avoiding cul-de-sacs and refining action sequences dynamically. |
| Tree-planner (Hu et al., 2023b) | VirtualHome | Success Rate, Executability, Goal Conditions Recall, Error, Correction Token Cost | 1. Plan sampling using LLM for prospective plans<br>2. Action tree construction<br>3. Grounded deciding with real-time observations | 1. Tree-Planner achieves a 1.29% improvement in Success Rate and a 53.29% reduction in token cost compared to ITERATIVE-PLANNER and a 40.52% reduction in error corrections.<br>2. Significantly enhances efficiency in LLM planning by reducing the number of LLM calls and interaction with the world model, and by minimizing input/output lengths and model sizes. |
| Thought of Search (Katz et al., 2024) | 24 Game, Mini Crosswords, BlocksWorld, PrOntoQA | Success Rate, Plan Time, Model Evaluation Calls | 1. Using LLMs to generate code for search components (successor functions and goal tests)<br>2. Search performed using BFS and DFS<br>3. Minimizes LLM calls for efficiency | Thought of Search achieves 100% accuracy across datasets with minimal LLM calls (1–2 calls per function). Significantly more efficient than existing LLM-based planning methods. |
| Chain-of-Symbols (Hu et al., 2023a) | Brick World, NLVR-based Manipulation, Natural Language Navigation, SPARTUN | Accuracy, Precision, Recall, Token Usage | 1. Translates spatial relationships to symbolic representation<br>2. Uses chained intermediate steps for efficient planning<br>3. Reduced redundancy in input tokens | 1. COS improves accuracy by 60.8% on Brick World tasks (from 31.8% to 92.6%) and reduces token usage by 65.8%.<br>2.Significantly enhances the correctness and efficiency of LLM planning by reducing unnecessary token usage by up to 65.8% and improving accuracy by up to 60.8% in complex spatial tasks. |
| Beyond A* (Lehnert et al., 2024) | Sokoban, Maze Navigation | Success Weighted by Cost (SWC), Improved Length Ratio (ILR), Optimal Plan Success Rate | 1. Trains Transformer to imitate A* search dynamics<br>2. Fine-tunes to generate shorter execution traces<br>3. Bootstraps from search dynamics for optimized plan generation | 1. Demonstrates novel training methods that integrate search dynamics into Transformer training, enhancing planning efficiency.<br>2. Achieves significant reductions in search steps (up to 26.8% fewer than A*) and improves task-solving performance on complex puzzles and navigation tasks. |
| LLM-DP (Dagan et al., 2023) | Alfworld | Success Rate, Average Episode Length | 1. LLM converts task descriptions into executable PDDL goals<br>2. Symbolic planner (BFS(f)) generates valid plans<br>3. Belief sampling to generate multiple world states for planning | 1. LLM-DP outperforms the ReAct baseline, achieving 96% success in Alfworld, with significantly fewer actions (13.16 vs. 18.69) per task and higher efficiency due to belief sampling and symbolic planning integration.<br>2. Reduces computational costs by decreasing the number of search steps required for planning, improving efficiency by up to 26.8% |
| DEPS (Wang et al., 2023b) | Alfworld | Success Rate, Average Episode Length | 1. LLM generates PDDL goal from task description<br>2. Sampling world beliefs and using symbolic planning<br>3. Plan generator (BFS(f)) for generating plans | 1. LLM-DP achieves 96% success in Alfworld, outperforming the ReAct baseline (53%) with fewer actions.<br>2.Reduces reliance on extensive LLM calls and minimizes human involvement by integrating automatic PDDL generation, thereby enhancing the efficiency of planning in dynamic environments. |
| PLaSma (Brahman et al., 2024) | COPLAN, VirtualHome | Human Evaluation: Coverage, Order, Overall Quality, Executability, Correctness | 1. Symbolic procedural knowledge distillation<br>2. Multi-task and task-specific distillation<br>3. Verifier-guided step-wise beam search<br>4. Constrained and counterfactual replanning tasks | 1. PlaSma enhances smaller models with procedural knowledge, outperforming GPT-3 in tasks like goal-based planning and counterfactual replanning. Achieves up to 93% success in counterfactual replanning.<br>2.Demonstrates efficiency by enabling smaller models to perform complex planning tasks with reduced computational costs. |
| KnowNo (Ren et al., 2023) | Simulated and Real Robot Tasks | Success Rate, Help Reduction, Task Success with Human Help, Prediction Set Size | 1. Use of Conformal Prediction (CP) for uncertainty calibration<br>2. Selection of multiple candidate actions<br>3. Minimizes human intervention based on prediction set size | 1. KNOWNO achieves 85% task success while reducing human help by 10–24%. Guarantees task success with a calibrated confidence level using CP.<br>2. Enhances efficiency and autonomy by reducing the need for human intervention, aligning uncertainty to ensure task success |
| AutoToS (Cao et al., 2024) | BlocksWorld, 24 Game, Sokoban, Mini Crosswords, PrOntoQA | Success Rate, Plan Accuracy, Number of Language Model Calls, Time to Solve | 1. Automates feedback process for search components 2. Uses soundness and completeness checks to refine the generated code 3.Incorporates both generic and domain-specific unit tests | 1. AutoToS achieves 100% accuracy across five domains with minimal human feedback. Significantly reduces the number of model calls and ensures soundness and completeness.<br>2.Improves efficiency by automating the feedback process, significantly reducing the number of LLM calls and human involvement. |