PDDL PLANNING WITH PRETRAINED LARGE LANGUAGE MODELS

Tom Silver[*,†], **Varun Hariprasad**[*,‡], **Reece Shuttleworth**[*,†],
Nishanth Kumar[†], **Tomás Lozano-Pérez**[†], **Leslie Pack Kaelbling**[†]
[†]Massachusetts Institute of Technology, [‡]Paul Laurence Dunbar High School

ABSTRACT

We study few-shot prompting of pretrained large language models (LLMs) towards solving PDDL planning problems. We are interested in two questions: (1) To what extent can LLMs solve PDDL planning problems on their own? (2) How and to what extent can LLMs be used to guide AI planners? Recent work by Valmeekam et al. (2022) presents negative evidence for (1) in the classic blocks world domain. We confirm this finding, but expand the inquiry to 18 domains and find more mixed results with a few clear successes. For (2), we propose a simple mechanism for using good-but-imperfect LLM outputs to aid a heuristic-search planner. We also find that the LLM performance is due not only to syntactic pattern matching, but also to its commonsense understanding of English terms that appear in the PDDL. Code: `https://tinyurl.com/llm4pddl`

1 INTRODUCTION

We are interested in the applicability of pretrained large language models (LLMs) (Brown et al., 2020; Chen et al., 2021; Chowdhery et al., 2022) to AI planning problems. Research in AI planning over the last two decades has led to powerful domain-independent planners (Helmert, 2006) and a wide variety of real-world applications (Sohrabi, 2019). AI planning problems are typically represented in PDDL (Fox & Long, 2003), a Lisp-like specification language. Given an initial state, goal, and (partial) transition model written in PDDL, AI planners are capable of generating actions that achieve the goal from the initial state. Do LLMs have similar capabilities? Can the commonsense understanding of an LLM be used to enhance the syntactic reasoning of an AI planner?

Towards answering these questions, we consider a few-shot prompting paradigm (Brown et al., 2020) where an LLM is queried with N examples of PDDL problems and solutions and 1 new problem from the same domain (Figure 1). The hope is that the LLM will complete this prompt with a valid solution to the new problem. There are several reasons to immediately doubt the viability of this enterprise. First, since LLMs are pretrained on natural language or general code, there is no guarantee that their outputs will even be syntactically valid PDDL. This lack of specificity to PDDL is a limitation that other learning-to-plan approaches do not have (Yoon et al., 2008; Jiménez et al., 2019; Karia & Srivastava, 2021; Gehring et al., 2022). Second, unlike planning-based approaches, the LLM does not have direct access to the transition model (PDDL operators), since it is prompted only with initial states, goals, and solutions[1]. Third, and perhaps most worryingly, recent work by Valmeekam et al. (2022) presents compelling evidence that "large language models still can't plan" in the classic blocks world PDDL domain.

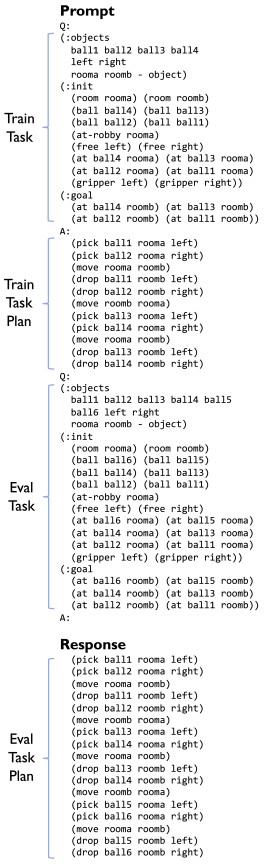

Figure 1: Few-shot LLM prompting for PDDL.

[*]Equal contribution. Correspondence: `tslvr@mit.edu`

[1]Preliminary experiments with including the operators (or English descriptions of them) in prompts did not yield clear improvements.

Despite these grounds for caution, we are motivated to continue for a few reasons. First, pretrainined LLMs have shown remarkable reasoning and problem-solving abilities in other contexts (Wei et al., 2022; Drori et al., 2022; Lewkowycz et al., 2022). Second, even though LLMs do not have direct transition model access, the example solutions may contain enough information for the LLM to generalize to new problems. For example, PDDL typically contains English names that a human, and perhaps the LLM, could use to understand the domain. The solutions may also exhibit a clear pattern that the LLM could potentially extrapolate, as in few-shot imitation learning (Cropper, 2019; Silver et al., 2020). Third, even if LLMs "still can't plan" on their own, they may still provide useful guidance to an AI planner.

In this work, we apply OpenAI's Codex LLM (Chen et al., 2021) to 18 PDDL domains: 17 from the Pyperplan benchmark collection (Alkhazraji et al., 2020) and 1 new PDDL domain. We choose to go "wide" in this pilot study in an effort to identify domains that merit future "deep" investigations. We find preliminary evidence to support the following claims:

1. In certain PDDL domains, LLMs are capable of solving some nontrivial problems.
2. However, in many other cases, LLMs cannot yet solve problems on their own.
3. LLMs are sensitive to the semantics of the English terms used in the PDDL problems.
4. LLMs can perform well in novel[2] PDDL domains.
5. Even when LLMs cannot solve problems, they can still be useful for planning.

Beyond our empirical findings, we make two additional contributions. (1) We propose a simple but effective mechanism to leverage the output of an LLM during planning. The main idea is to use the LLM output to initialize the queue of a greedy best-first search planner. (2) We contribute[3] an open-source codebase that can be used to benchmark the planning performance of future LLMs, and to develop new techniques for combining LLMs and planners.

2 Related Work

There has been considerable recent interest in leveraging LLMs for decision making (Collins et al., 2022; Cohen et al., 2021; Simon & Muise, 2022). For example, Sharma et al. (2022) fine-tune a pretrained LLM to generate sequences of natural language instructions for robotic tasks in the ALFRED household environment (Shridhar et al., 2020). Li et al. (2022) consider a similar fine-tuning strategy in the VirtualHome environment (Puig et al., 2018) and additionally propose an active data gathering process. Huang et al. (2022a) also consider VirtualHome, but with few-shot prompting for instruction generation. They propose a mechanism for constraining the LLM output to feasible action sequences, which we adopt in this work (Section 4). Huang et al. (2022b) incorporate feedback into the prompts to recover from skill execution failures. Ahn et al. (2022) use LLM embeddings and pretrained skill value functions to map natural language instructions to skills in a large suite of robotics tasks.

Most relevant is recent work by Valmeekam et al. (2022), who consider solving PDDL blocks world problems with LLM few-shot prompting and propose blocks world as a benchmark for improving decision-making with LLMs. Their "plan generalization" setting is the most similar to ours. One difference is that we prompt with the original PDDL syntax, rather than domain-specific natural language encodings of the PDDL. Accordingly, we use Codex — a version of GPT-3 fine-tuned on general code — instead of GPT-3 itself.[4] Valmeekam et al. (2022) come to a largely negative conclusion about LLM planning ability ("large language models still can't plan"). In looking beyond blocks world, we find evidence to support a more optimistic view. We also show that LLMs can be useful to guide planning, even in domains where LLMs alone struggle to find complete plans on their own.

[2]Since LLMs were pretrained on a large corpus of natural language and code, it is likely that some of the benchmark PDDL domains that we study in this work appeared in pretraining data. For this reason, we also evaluate on a new domain ("Dressed") that we created after the LLMs were pretrained.

[3]`https://tinyurl.com/llm4pddl`

[4]However, preliminary experiments suggest that performance with GPT-3 is similar.

3 BACKGROUND

We begin with a brief review of PDDL planning and LLMs. See Geffner & Bonet (2013) and Brown et al. (2020) respectively for more detailed introductions.

3.1 PDDL PLANNING

We consider AI planning in a setting with discrete and fully-observable states, finite actions, deterministic transitions, and goal-based problems with no other extrinsic feedback. In each state $s \in \mathcal{S}$, the agent can select an action $a \in \mathcal{A}$ from the set of *applicable actions* $\mathcal{A}(s) \subseteq \mathcal{A}$. A known *transition model* $F : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ determines the next state after an action is selected. The transition model is partial: $F(s, a)$ is defined if and only if $a \in \mathcal{A}(s)$. A *problem* consists of an initial state $s_0 \in \mathcal{S}$ and set of goal states $g \subseteq \mathcal{S}$. A *solution* to a problem is a plan $\overline{a} = (a_0, \ldots, a_{n-1})$ that results in a goal state, that is, $s_{i+1} = F(s_i, a_i)$ for all $0 \leq i < n$ and $s_n \in g$. A *domain* is characterized by a state space, an action space, a transition model, and a problem distribution. In this work, our objective is to maximize the expected number of problems solved from a domain's problem distribution within a wall-clock timeout. We are thus concerned with satisficing, not optimal, planning; we wish to find *any* solution, not necessarily a shortest one.

We consider domains encoded in the STRIPS subset of PDDL (Fox & Long, 2003). A state s is represented by a set of *objects* and a set of *ground atoms*. Each ground atom (e.g., `(on a b)`) consists of a predicate (`on`) and ordered object arguments (`a` and `b`). The object set is constant for a problem but changes between problems. Objects can also be typed (e.g., `a - block`). A problem goal g is also represented by a set of ground atoms and the goal is achieved in state s if $g \subseteq s$. An action a is represented by a *ground operator* (e.g., `(unstack a b)`), which consists of an operator (`unstack`) and ordered object arguments (`a` and `b`). A ground operator a has *preconditions* — a set of ground atoms — and is applicable ($a \in \mathcal{A}(s)$) if the preconditions are a subset of s. The ground operator also has *add effects* and *delete effects*, each a set of ground atoms, which define the transition model: the add effects are added to the state, and the delete effects are removed from it. PDDL specifies a Lisp-like syntax for each of these representations.

An AI planner consumes a PDDL domain and problem as input and produces a solution. Most state-of-the-art (SOTA) planners (Helmert, 2006; Hoffmann, 2001) are variants of heuristic-based forward search, where the PDDL encodings are used to automatically derive a heuristic function. In this work, we primarily use the Pyperplan planner with greedy best-first search (GBFS) and the hFF heuristic (Alkhazraji et al., 2020). Pyperplan is not SOTA, but we prefer it in this work due to its clean implementation and ease of extensibility. We also report results for the SOTA Fast Downward planner for comparison (Helmert, 2006).

3.2 LARGE LANGUAGE MODELS (LLMS)

A large language model (LLM) is a model with a large number of parameters (on the order of 10^{11}) that is trained to represent a probability distribution over a large corpus of text (e.g. the webtext corpora (Gao et al., 2020)). Though these models are trained simply to predict the likelihood of text conditioned on some previous context, they exhibit impressive performance on a variety of downstream tasks like 3-digit arithmetic and word problems (Lewkowycz et al., 2022). An LLM takes a text query as input and predicts probabilities over following text. We use only the most likely output[5]. In this work, we use the Codex LLM (Chen et al., 2021).

4 SOLVING PDDL PROBLEMS WITH FEW-SHOT PROMPTING

We now describe our approaches for using pretrained LLMs to solve PDDL problems. All approaches use few-shot prompting, where N training problems and solutions are generated once offline per domain and prepended to each new evaluation problem from the domain that we wish to solve. The LLM is then queried with this complete prompt and the resulting output is used to construct a plan in one of several ways. We detail each of these steps in the following subsections.

[5]Sampling multiple LLM outputs (up to 10 with various temperatures) did not yield clear improvements.

4.1 PROMPT CONSTRUCTION

To create examples for few-shot prompting, we reserve N prompting problems from the domain distribution and use a planner[6] to generate one solution per problem. In practice, we select the smallest[7] N problems for prompting from a benchmark set. Following previous work (e.g., (Kojima et al., 2022)), we construct a prompt prefix by sequencing together **Q:** `<problem>` and **A:** `<solution>` for each of the N examples. The problem and solution are encoded in PDDL syntax. For each new evaluation problem in the domain, we independently create a prompt by concatenating the prefix and **Q:** `<new problem>` **A:** . See Figure 1 for a simple example.

4.2 LLM OUTPUTS TO PDDL PLANS

Given a prompt, we query the LLM and record its response until **Q:** is encountered (the stop token) or a maximum length is exceeded (the context window). We then attempt to parse the response into a PDDL plan. Our parser is relatively naive and assumes that the response is a sequence of actions in the form `(<operator> <object> <object> ...)`. Since the LLM is unconstrained, it is certainly possible for the response to contain syntax errors, e.g., invalid names or operators with invalid arguments. It is also possible that the response represents a plan that violates the preconditions of the action at some step or does not reach the goal.

We consider two strategies for dealing with malformed LLM outputs: No Validation and Soft Validation. With **No Validation**, we simply skip over any invalid action over during output parsing. With **Soft Validation**, we use the technique of Huang et al. (2022a) to constrain the outputs to be valid PDDL plans. In this autoregressive prompting technique, rather than generating an entire plan all at once, we generate one action at a time. After each closed-parentheses expression is output by the LLM, we check to see if it already represents a valid action. If not, we find the valid action that is "closest" to the generated expression, where distance is defined via cosine similarity in a pretrained embedding space. We use the Sentence-BERT embedding model (`paraphrase-MiniLM-L6-v2`) from Hugging Face in experiments (Reimers & Gurevych, 2019). After each valid action is obtained, we first check to see if the current plan is a solution. Otherwise, the valid action is appended to the prompt and the LLM is re-queried.

4.3 USING LLM OUTPUTS TO GUIDE A PLANNER

The above techniques are designed to address the question of whether LLMs can solve PDDL problems on their own. We are also interested in the questions of how and to what extent LLMs can be used to guide a PDDL planner. To this end, we propose a simple approach — **LLM Plan Guidance** — for using the LLM output with a search-based planner. Recall that heuristic search algorithms maintain a priority queue of nodes, where each node contains a partial plan and a corresponding state sequence. Given the LLM output, we first parse it into a plan $\overline{a} = (a_0, \ldots, a_{n-1})$ by either the autoregressive or non-autoregressive technique described in Section 4.2. We then use the initial state s_0 and the transition model F to generate the corresponding state sequence $(s_0, \ldots, s_n)$. Then for each step i where $0 \leq i < n$, we add to the initialized priority queue a node with the partial plan $(a_0, \ldots, a_i)$ and the corresponding state sequence $(s_0, \ldots, s_{i+1})$. We use the standard priority function to order these nodes. For example, in GBFS, the priority is equal to the heuristic.

To motivate LLM Plan Guidance, consider first the case where the LLM output already encodes a solution. Assuming a well-behaved heuristic, GBFS will immediately pop the full plan $\overline{a}$ node from the priority queue and return the solution before expanding any other nodes. Consider instead a case where the LLM outputs an almost complete plan, but it is missing one or a few actions at the end. If the heuristic is a good approximation of the true cost-to-go, GBFS would again pop the full plan and continue searching from the end, likely exploring fewer nodes than it would starting from the initial state. The LLM would provide especially useful guidance in the case where its output is sufficient for the planner to escape from some early local optima in the heuristic landscape. The extent to which this LLM guidance benefits the planner is ultimately an empirical question; to answer it, we now turn to experiments.

[6] We use Fast Downward with the `lama-first` configuration to generate these example solutions.

[7] Problem length is measured by counting ground atoms in the initial state and goal.

5 EXPERIMENTS AND RESULTS

5.1 DOMAINS

We use the Pyperplan domains and problems for our main experiments (Alkhazraji et al., 2020). Pyperplan includes 21 domains. We exclude 4 domains whose operators vary per problem (Airplane, Openstack, Parcprinter, and PSR) and use the remaining 17 domains. We sort the problems by length and use the first $N = 2$ problems for prompting and the next 10 problems for evaluation.

Since the Codex LLM was pretrained on open-source code (Chen et al., 2021), and since the Pyperplan benchmarks are open-source, it is possible that some or all of these domains appeared in the pretraining data [8]. To mitigate the possibility that our results are due to memorization, we designed a new "Dressed" PDDL domain that, to the best of knowledge, is qualitatively dissimilar to existing PDDL domains. This domain involves planning for one or more people to get dressed for a casual or formal event. Problems are procedurally generated and vary in the number of people, events, and clothing. We again use $N = 2$ small train problems and 10 larger evaluation problems.

5.2 APPROACHES

We now describe the approaches and baselines that we evaluate on all domains.

5.2.1 OPEN-LOOP APPROACHES

Our first set of "open-loop" approaches are designed to address the question of whether LLMs can plan on their own. These approaches include:

- **Soft Validation:** Our main approach with autoregressive prompting as described in Section 4.2.
- **No Validation:** Same, but *without* autoregressive prompting.
- **Ablate Names:** Same as No Validation, but each operator, object, type, and predicate name is randomly replaced with one of the 10,000 most common English words (Price, 2022).
- **Random Actions:** Selects a random applicable action for up to n steps or until the goal is reached, where n is equal to the length of the Soft Validation output per problem.

5.2.2 PLANNING-BASED APPROACHES

Our second set of approaches are designed to address the question of whether LLMs can be useful in guiding a heuristic search planner. We use Pyperplan's GBFS search with the hFF heuristic unless otherwise noted with a 300 second planning timeout (LLM query time not included). These approaches include:

- **LLM Plan Guidance:** Our main approach (with autoregressive prompting) where the LLM output is used to initialize the GBFS priority queue (Section 4.3).
- **LLM Plan Guidance No Val:** Same, but without autoregressive prompting.
- **Random Plan Guidance:** The random plans generated by the Random Action baseline from Section 5.2.1 are used to initialize the queue instead of the LLM outputs.
- **Pure Planning:** The Pyperplan planner with no additional guidance.
- **Fast Downward:** The Fast Downward planner with the `lama-first` configuration (Helmert, 2006). For space, these results are deferred to the appendix.

5.3 EXPERIMENTAL SETUP

All experimental results are reported over 5 random seeds. Variation between seeds is driven by nondeterministism in Pyperplan's implementation of hFF, random action selection in the baselines, random object renaming in the Ablate Names ablation, and small variations in processor speed. All LLM-based approaches use the `code-davinci-002` Codex LLM (Chen et al., 2021). Queries are made through the OpenAI API with temperature 0.0. Experiments were run on Ubuntu 18.04 using 4 CPU cores of an Intel Xeon Platinum 8260 processor.

[8] Through prompt engineering, we have seen strong indications that Codex was trained on at least Blocks. For example, prompting with the beginning of the PDDL domain file, Codex will complete the rest of the file.

5.4 Results and Analysis

Table 1 reports the fraction of the 10 evaluation problems solved by each of the open-loop (non-planning) approaches in each of the domains, averaged over 5 seeds. Our first conclusion is that **in certain PDDL domains, LLMs are capable of solving some nontrivial problems**. Figure 1 shows one case where the LLM generalizes from a Gripper prompting problem with 4 balls to an evaluation problem with 6 balls. This is the easiest case; the LLM also generalizes to all evaluation problems in the set, including the largest with 26 balls and 77 required actions. The Gripper domain is also the simplest in terms of the extrapolation required, since all problems can be solved by a sequence of repeating pick, pick, move, drop, drop actions. Other domains like Miconic, Rovers, Scanalyzer, TPP, and Woodworking require considerably more extrapolation, and in each of these cases, the Soft Validation approach achieves nontrivial performance.

	Soft Val	No Val	Ablate Names	Random Actions
Blocks	0.00	0.00	0.00	0.00
Depot	0.00	0.00	0.00	0.00
Dressed	1.00	0.80	0.00	0.00
Elevators	0.00	0.00	0.00	0.00
Freecell	0.00	0.00	0.00	0.12
Gripper	1.00	1.00	0.98	0.00
Logistics	0.00	0.00	0.00	0.00
Miconic	0.20	0.00	0.00	0.10
Movie	1.00	1.00	0.08	0.00
Pegsol	0.00	0.00	0.00	0.00
Rovers	0.10	0.00	0.00	0.04
Satellite	0.00	0.00	0.00	0.00
Scanalyzer	0.20	0.10	0.10	0.02
Sokoban	0.00	0.00	0.00	0.00
TPP	0.30	0.20	0.18	0.00
Transport	0.00	0.00	0.00	0.04
Woodworking	0.20	0.00	0.00	0.00
Zenotravel	0.00	0.00	0.00	0.00
Average	**0.222**	**0.172**	**0.074**	**0.018**

Table 1: Mean fraction of tasks solved by the open-loop approaches. See Table 3 for standard deviations.

Our second conclusion is that **in many other domains, LLMs cannot yet solve problems on their own**. This is evident from the zero entries in the Soft Validation column, including in the Blocks domain, which confirms the findings of prior work (Valmeekam et al., 2022). Upon inspection, we find that gross syntax errors in the output of the LLM are very rare, but mistaken actions are common, and occur both in the operators and object arguments. We also find that the LLM struggles with certain spatial concepts, e.g., understanding that a block tower needs to be built from the bottom up, or that an elevator should move up to reach higher floors and down to reach lower floors.

The difference in performance between No Validation and Soft Validation supports previous findings (Huang et al., 2022a) that constraining the LLM output to valid plans can lead to substantially better performance. However, it is worth noting that the autoregressive prompting required by Soft Validation dramatically increases the approach run time, since there is significant latency in querying the LLM. We also conducted preliminary experiments with the dynamic prompting method proposed by Huang et al. (2022a), but did not find significant improvements in our setting, likely due to a limited number of prompting problems from which we could select.

By comparing No Validation to Ablate Names, we see a sharp decline in performance, suggesting that **LLMs are sensitive to the semantics of the English terms used in the PDDL problems.** If the LLMs were instead using syntactic pattern matching to generalize from the prompts, we would expect to see minimal performance decline when terms are substituted. Additional experiments with random strings, rather than English words, supported the same conclusion.

The perfect evaluation performance of Soft Validation in the Dressed domain supports the claim that **LLMs can perform well in novel domains**, not just in domains that may have existed in some form in the LLM pretraining corpus.

	LLM Plan Guidance			LLM No Val Plan Guidance			Random Plan Guidance		
	created	expanded	success	created	expanded	success	created	expanded	success
Blocks	8.09	-10.72	0.00	25.37	14.43	0.00	58.10	5.01	0.00
Depot	-61.94	-60.11	17.39	-47.84	-48.88	-8.70	-2.16	-3.33	17.39
Dressed	-99.96	-99.95	21.95	-99.96	-99.95	21.95	-17.73	-14.06	-2.44
Elevators	-18.81	-25.48	0.00	-10.06	-12.49	0.00	-5.57	-11.38	0.00
Freecell	-13.89	3.86	4.17	3.73	17.19	4.17	-12.69	39.70	4.17
Gripper	-98.44	-99.74	0.00	-98.44	-99.74	0.00	-40.06	-34.95	0.00
Logistics	-34.82	-45.08	0.00	-15.17	-18.42	0.00	-1.31	-16.61	0.00
Miconic	-19.17	-36.32	0.00	-30.83	-48.83	0.00	-3.98	-42.63	0.00
Movie	-71.88	-87.50	0.00	-71.88	-62.50	0.00	-67.38	-53.75	0.00
Pegsol	44.77	52.13	-4.35	24.34	25.89	6.52	16.75	19.61	2.17
Rovers	-19.73	-16.78	0.00	-29.69	-29.23	2.04	-62.73	-65.51	2.04
Satellite	-54.32	-64.51	0.00	-70.37	-70.77	0.00	-34.96	-38.21	-4.00
Scanalyzer	-14.86	-16.71	-2.44	5.28	4.52	2.44	-48.28	-52.92	-4.88
Sokoban	-19.95	-19.72	0.00	-5.46	-5.94	0.00	-16.55	-17.15	-2.78
TPP	-89.55	-88.75	-2.00	-7.44	-4.85	0.00	-98.25	-98.01	0.00
Transport	-0.44	-2.63	0.00	9.86	12.36	0.00	24.04	-20.06	0.00
Woodworking	312.42	347.66	0.00	0.27	-0.56	0.00	0.39	-0.92	0.00
Zenotravel	0.86	-9.54	0.00	-12.76	-22.26	0.00	-7.05	-27.99	0.00

Table 2: Performance of the planning guidance approaches. All numbers are a percentage increase or decrease relative to the Pure Planning baseline. Green represents $\geq$10% better, red represents $\geq$10% worse, and black represents $<$10% change. The first two inner columns report the number of nodes "created" and "expanded." The third reports the fraction of tasks solved. See text for details and see the appendix for the absolute metrics, standard deviations, and additional results.

Table 2 reports three metrics for each of the planning-based approaches: the fraction of the 10 evaluation problems solved, the average number of nodes created during planning, and the number of nodes expanded. All metrics are reported as a percentage increase or decrease relative to the Pure Planning approach. For fair comparison, node creation and expansion results are averaged over only the problems solved by all planning-based approaches. These results suggest that **even when LLMs cannot solve problems, they can still be useful for planning**. For example, in the Depot, Elevators, Logistics, Satellite, and Sokoban domains, Soft Validation does not solve any problems alone, but LLM Plan Guidance strictly improves over Pure Planning. By comparing LLM Plan Guidance and Random Plan Guidance, we also confirm that the improved performance is not simply due to initializing the queue with arbitrary partial plans; the LLM is providing nontrivial guidance.

Limitations. Although these results are encouraging, note that SOTA AI planning remains far more efficient than LLM Plan Guidance in terms of wall-clock time. There is high latency in querying the LLM, and there is a marked gap between the Pyperplan planner and the SOTA Fast Downward planner (see Table 7). Another limitation of current LLMs is their fixed-size context window for prompts and responses (4096 tokens for Codex). This context window can be a major bottleneck for PDDL. We experimented with prompt compression techniques, but found that using $N > 2$ examples consistently degraded overall performance because of exceeded context windows. We do see that $N = 2$ is better than $N = 1$, which suggests that if the context window could be increased to accommodate larger N, results may improve further. Despite these caveats, the results we present here are promising and motivate further research in using LLMs for PDDL planning.

6 Discussion and Conclusion

In this work, we investigated the extent to which LLMs can plan on their own and be used to guide planning in PDDL domains. We found that LLMs alone can solve some nontrivial PDDL problems, but fail to solve many others. However, even in problems that LLMs cannot solve, we showed that their outputs can be useful in guiding a heuristic-search planner. There remains a large gap between SOTA and LLM-based planners, but our results suggest that further research is well-motivated. We hope that our open-source code can serve as a benchmark for progress in planning with LLMs and can help foster collaboration between the LLM and PDDL planning communities.

ACKNOWLEDGEMENTS

We thank Gabe Grand, Michael Katz, Wenlong Huang, and the FMDM reviewers for helpful feedback on an earlier draft. We gratefully acknowledge support from NSF grant 2214177; from AFOSR grant FA9550-22-1-0249; from ONR grant N00014-18-1-2847; from the MIT-IBM Watson Lab; and from the MIT Quest for Intelligence. Tom and Nishanth are supported by NSF Graduate Research Fellowships. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

REFERENCES

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

Yusra Alkhazraji, Matthias Frorath, Markus Grützner, Malte Helmert, Thomas Liebetraut, Robert Mattmüller, Manuela Ortlieb, Jendrik Seipp, Tobias Springenberg, Philip Stahl, and Jan Wülfing. Pyperplan. `https://doi.org/10.5281/zenodo.3700819`, 2020. URL `https://doi.org/10.5281/zenodo.3700819`.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Vanya Cohen, Geraud Nangue Tasse, Nakul Gopalan, Steven James, Matthew Gombolay, and Benjamin Rosman. Learning to follow language instructions with compositional policies. *arXiv preprint arXiv:2110.04647*, 2021.

Katherine M Collins, Catherine Wong, Jiahai Feng, Megan Wei, and Joshua B Tenenbaum. Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks. *arXiv preprint arXiv:2205.05718*, 2022.

Andrew Cropper. Playgol: Learning programs through play. *arXiv preprint arXiv:1904.08993*, 2019.

Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, et al. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32):e2123433119, 2022.

Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Hector Geffner and Blai Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141, 2013.

Clement Gehring, Masataro Asai, Rohan Chitnis, Tom Silver, Leslie Kaelbling, Shirin Sohrabi, and Michael Katz. Reinforcement learning for classical planning: Viewing heuristics as dense reward generators. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pp. 588–596, 2022.

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

Jörg Hoffmann. Ff: The fast-forward planning system. *AI magazine*, 22(3):57–57, 2001.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning (ICML)*, 2022a.

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

Sergio Jiménez, Javier Segovia-Aguas, and Anders Jonsson. A review of generalized planning. *The Knowledge Engineering Review*, 34, 2019.

Rushang Karia and Siddharth Srivastava. Learning generalized relational heuristic networks for model-agnostic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8064–8073, 2021.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

Eric Price. 10,000 most common english words. `https://www.mit.edu/~ecprice/wordlist.10000`, 2022. Accessed: 2022-09-27.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `http://arxiv.org/abs/1908.10084`.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1713–1726, 2022.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Tom Silver, Kelsey R Allen, Alex K Lew, Leslie Pack Kaelbling, and Josh Tenenbaum. Few-shot bayesian imitation learning with logical program policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10251–10258, 2020.

Nisha Simon and Christian Muise. Tattletale: Storytelling with planning and large language models. In *ICAPS Workshop on Scheduling and Planning Applications woRKshop*. 2022.

Shirin Sohrabi. Ai planning for enterprise: Putting theory into practice. In *IJCAI*, pp. 6408–6410, 2019.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). *arXiv preprint arXiv:2206.10498*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Sungwook Yoon, Alan Fern, and Robert Givan. Learning control knowledge for forward search planning. *Journal of Machine Learning Research*, 9(4), 2008.

A ADDITIONAL RESULTS

Here we report our full set of results with both means and standard deviations. Table 3 includes all standard (non-planning) approaches. Table 4 reports LLM Plan Guidance results. Table 5 reports Random Plan Guidance results. Table 6 reports Pure Planning results. Table 7 reports Fast Downward results. Table 8 reports LLM Plan Guidance No Autoregress results.

	Soft Validation	No Validation	Ablate Names	Random Actions
Blocks	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Depot	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Dressed	1.00 (0.00)	0.80 (0.00)	0.00 (0.00)	0.00 (0.00)
Elevators	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Freecell	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.12 (0.04)
Gripper	1.00 (0.00)	1.00 (0.00)	0.98 (0.04)	0.00 (0.00)
Logistics	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Miconic	0.20 (0.00)	0.00 (0.00)	0.00 (0.00)	0.10 (0.09)
Movie	1.00 (0.00)	1.00 (0.00)	0.08 (0.07)	0.00 (0.00)
Pegsol	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Rovers	0.10 (0.00)	0.00 (0.00)	0.00 (0.00)	0.04 (0.05)
Satellite	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Scanalyzer	0.20 (0.00)	0.10 (0.00)	0.10 (0.00)	0.02 (0.04)
Sokoban	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
TPP	0.30 (0.00)	0.20 (0.00)	0.18 (0.04)	0.00 (0.00)
Transport	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.04 (0.05)
Woodworking	0.20 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Zenotravel	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Average	**0.222 (0.0)**	**0.172 (0.0)**	**0.074 (0.008)**	**0.018 (0.015)**

Table 3: Mean (standard deviation) fraction of evaluation tasks solved by open-loop (non-planning) approaches, averaged over 5 seeds.

	LLM Plan Guidance		
	created	expanded	success
Blocks	101.32 (14.91)	45.66 (8.34)	1.00 (0.00)
Depot	1639.02 (662.88)	636.86 (273.57)	0.54 (0.10)
Dressed	18.50 (0.00)	0.70 (0.00)	1.00 (0.00)
Elevators	454.06 (18.83)	32.34 (2.12)	1.00 (0.00)
Freecell	97.32 (11.32)	32.86 (8.12)	0.50 (0.00)
Gripper	51.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Logistics	140.82 (1.83)	15.74 (0.26)	1.00 (0.00)
Miconic	215.18 (1.12)	24.02 (1.53)	1.00 (0.00)
Movie	9.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Pegsol	2583.02 (2226.26)	2062.56 (1855.46)	0.88 (0.07)
Rovers	563.88 (35.34)	51.98 (4.76)	0.98 (0.04)
Satellite	1052.88 (99.57)	19.60 (2.47)	1.00 (0.00)
Scanalyzer	440.96 (43.12)	36.48 (4.32)	0.80 (0.06)
Sokoban	3508.88 (495.29)	2935.92 (405.77)	0.72 (0.04)
TPP	856.94 (143.11)	136.26 (30.93)	0.98 (0.04)
Transport	191.84 (28.27)	42.22 (8.43)	1.00 (0.00)
Woodworking	4183.52 (2651.84)	561.90 (365.81)	1.00 (0.00)
Zenotravel	485.00 (99.07)	25.60 (6.76)	1.00 (0.00)
Average	**921.841 (362.931)**	**370.15 (165.481)**	**0.911 (0.019)**

Table 4: Means (standard deviations) for the LLM Plan Guidance approach. See text for details.

	Random Plan Guidance		
	created	expanded	success
Blocks	148.20 (24.68)	53.70 (14.26)	1.00 (0.00)
Depot	4213.68 (2455.50)	1543.42 (994.07)	0.54 (0.05)
Dressed	34612.94 (4065.65)	1229.24 (134.15)	0.80 (0.06)
Elevators	528.10 (69.08)	38.46 (11.13)	1.00 (0.00)
Freecell	98.68 (37.17)	44.20 (34.88)	0.50 (0.00)
Gripper	1965.74 (140.68)	249.80 (14.26)	1.00 (0.00)
Logistics	213.24 (6.25)	23.90 (0.70)	1.00 (0.00)
Miconic	255.60 (4.31)	21.64 (0.67)	1.00 (0.00)
Movie	10.44 (0.48)	3.70 (0.22)	1.00 (0.00)
Pegsol	2083.10 (780.12)	1621.66 (639.95)	0.94 (0.05)
Rovers	261.82 (54.11)	21.54 (5.85)	1.00 (0.00)
Satellite	1499.24 (527.00)	34.12 (15.65)	0.96 (0.08)
Scanalyzer	267.88 (105.10)	20.62 (9.17)	0.78 (0.10)
Sokoban	3658.06 (478.95)	3030.10 (392.63)	0.70 (0.00)
TPP	143.16 (73.32)	24.10 (15.95)	1.00 (0.00)
Transport	239.00 (26.52)	34.66 (6.07)	1.00 (0.00)
Woodworking	1018.36 (50.12)	124.36 (6.32)	1.00 (0.00)
Zenotravel	446.98 (63.02)	20.38 (3.86)	1.00 (0.00)
Average	**2870.234 (497.892)**	**452.2 (127.766)**	**0.901 (0.019)**

Table 5: Means (standard deviations) for the Random Plan Guidance approach. See text for details.

	Pure Planning		
	created	expanded	success
Blocks	93.74 (4.65)	51.14 (2.81)	1.00 (0.00)
Depot	4306.74 (1132.99)	1596.62 (433.75)	0.46 (0.08)
Dressed	42073.44 (3142.31)	1430.38 (118.71)	0.82 (0.04)
Elevators	559.24 (53.73)	43.40 (6.61)	1.00 (0.00)
Freecell	113.02 (20.83)	31.64 (18.51)	0.48 (0.04)
Gripper	3279.50 (0.00)	384.00 (0.00)	1.00 (0.00)
Logistics	216.06 (1.71)	28.66 (0.12)	1.00 (0.00)
Miconic	266.20 (1.36)	37.72 (1.24)	1.00 (0.00)
Movie	32.00 (0.00)	8.00 (0.00)	1.00 (0.00)
Pegsol	1784.18 (626.01)	1355.80 (499.54)	0.92 (0.04)
Rovers	702.46 (120.01)	62.46 (12.28)	0.98 (0.04)
Satellite	2304.94 (455.30)	55.22 (16.13)	1.00 (0.00)
Scanalyzer	517.90 (34.39)	43.80 (3.13)	0.82 (0.07)
Sokoban	4383.28 (356.86)	3657.18 (294.58)	0.72 (0.04)
TPP	8198.14 (501.09)	1211.42 (93.42)	1.00 (0.00)
Transport	192.68 (10.74)	43.36 (2.80)	1.00 (0.00)
Woodworking	1014.38 (44.49)	125.52 (5.84)	1.00 (0.00)
Zenotravel	480.86 (90.42)	28.30 (6.49)	1.00 (0.00)
Average	**3917.709 (366.494)**	**566.368 (84.22)**	**0.9 (0.019)**

Table 6: Means (standard deviations) for the Pure Planning approach. See text for details.

	Fast Downward		
	created	expanded	success
Blocks	39.50 (0.00)	38.50 (0.00)	1.00 (0.00)
Depot	595.70 (0.00)	595.30 (0.00)	0.90 (0.00)
Dressed	53.50 (0.00)	52.40 (0.00)	1.00 (0.00)
Elevators	96.60 (0.00)	95.60 (0.00)	1.00 (0.00)
Freecell	20.90 (0.00)	20.50 (0.00)	1.00 (0.00)
Gripper	75.50 (0.00)	74.50 (0.00)	1.00 (0.00)
Logistics	61.50 (0.00)	60.50 (0.00)	1.00 (0.00)
Miconic	82.80 (0.00)	81.80 (0.00)	1.00 (0.00)
Movie	9.00 (0.00)	8.00 (0.00)	1.00 (0.00)
Pegsol	623.70 (0.00)	455.70 (0.00)	1.00 (0.00)
Rovers	61.80 (0.00)	60.90 (0.00)	1.00 (0.00)
Satellite	36.80 (0.00)	36.00 (0.00)	1.00 (0.00)
Scanalyzer	43.50 (0.00)	42.90 (0.00)	1.00 (0.00)
Sokoban	4264.00 (0.00)	3685.90 (0.00)	0.90 (0.00)
TPP	196.10 (0.00)	195.20 (0.00)	1.00 (0.00)
Transport	48.90 (0.00)	47.90 (0.00)	1.00 (0.00)
Woodworking	833.20 (0.00)	485.60 (0.00)	1.00 (0.00)
Zenotravel	53.30 (0.00)	52.30 (0.00)	1.00 (0.00)
Average	**399.794 (0.0)**	**338.306 (0.0)**	**0.989 (0.0)**

Table 7: Means (standard deviations) for the SOTA Fast Downward planner. See text for details.

	LLM Plan Guidance No Validation		
	created	expanded	success
Blocks	117.52 (9.28)	58.52 (5.89)	1.00 (0.00)
Depot	2246.22 (577.72)	816.22 (226.15)	0.42 (0.04)
Dressed	18.80 (0.00)	0.70 (0.00)	1.00 (0.00)
Elevators	502.98 (10.81)	37.98 (1.49)	1.00 (0.00)
Freecell	117.24 (28.71)	37.08 (19.66)	0.50 (0.00)
Gripper	51.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Logistics	183.28 (1.81)	23.38 (0.24)	1.00 (0.00)
Miconic	184.14 (0.58)	19.30 (0.60)	1.00 (0.00)
Movie	9.00 (0.00)	3.00 (0.00)	1.00 (0.00)
Pegsol	2218.52 (1387.06)	1706.84 (1088.51)	0.98 (0.04)
Rovers	493.92 (72.36)	44.20 (8.71)	1.00 (0.00)
Satellite	682.94 (121.39)	16.14 (1.92)	1.00 (0.00)
Scanalyzer	545.22 (48.98)	45.78 (4.45)	0.84 (0.05)
Sokoban	4144.00 (389.72)	3439.78 (343.55)	0.72 (0.04)
TPP	7587.92 (537.90)	1152.66 (67.44)	1.00 (0.00)
Transport	211.68 (22.10)	48.72 (4.62)	1.00 (0.00)
Woodworking	1017.08 (33.38)	124.82 (4.52)	1.00 (0.00)
Zenotravel	419.50 (44.83)	22.00 (3.06)	1.00 (0.00)
Average	**1152.831 (182.591)**	**422.118 (98.934)**	**0.914 (0.009)**

Table 8: Means (standard deviations) for the LLM Plan Guidance No Validation approach. See text for details.