Cost-Augmented Monte Carlo Tree Search for LLM-Assisted Planning

Zihao Zhang and Fei Liu

Computer Science Department, Emory University

{zihao.zhang, fei.liu}@emory.edu

Abstract

While LLMs excel at open-ended reasoning, they often struggle with cost-sensitive planning, either treating all actions as having equal cost or failing to stay within strict budgets. In this paper, we introduce Cost-Augmented Monte Carlo Tree Search (CATS), a novel approach that brings explicit cost-awareness into LLM-guided planning. Tight cost constraints push the planner to quickly identify infeasible solutions, while looser constraints encourage optimization for minimal cost. We benchmark top LLMs such as GPT-4.1, Claude-3.7-Sonnet, and DeepSeek-R1, against our CATS planner to evaluate their performance in cost-sensitive scenarios. Our experiments suggest that raw LLMs such as GPT-4.1 often falter under tight budgets, whereas CATS consistently delivers strong performance, achieving higher task success rates and better cost efficiency. CATS provides an effective solution for budget-aware decision-making by combining the reasoning power of LLMs with structured search.

1 Introduction

Planning is a cornerstone of real-world decision-making, yet most research on LLM-assisted planning overlooks a critical factor: cost (Huang et al., 2024; Li et al., 2024; Wei et al., 2025). Whether it is travel planning with budget constraints, scheduling meetings with transition times, or strategic resource allocation, cost considerations are paramount (Xie et al., 2024; Zheng et al., 2024; Kambhampati et al., 2024). Some constraints are hard, e.g., missing a train departure, while others are soft, e.g., slightly exceeding a budget, but both demand cost-aware solutions. Existing LLM-based planners often assume all actions have the same cost, leading to suboptimal solutions. Our work bridges this gap by introducing cost-augmented planning, where models explicitly optimize for both task success and cost efficiency.

Recent advances have demonstrated LLMs' impressive reasoning capabilities, particularly in code generation, mathematical problem-solving, and logical reasoning (Shao et al., 2024; Ke et al., 2025; Hao et al., 2025). These breakthroughs suggest that LLMs could, in principle, handle constrained planning tasks by leveraging their emergent world understanding and step-by-step reasoning. However, while LLMs excel in open-ended reasoning, their ability to adhere to strict cost constraints remains understudied (Kambhampati et al., 2024). Can LLMs natively solve cost-sensitive planning problems, or do they require algorithmic enhancements to resolve constraints effectively? Our work investigates this question, testing both raw LLM performance and hybrid approaches that combine LLMs with structured search.

Monte Carlo Tree Search (MCTS) has a proven track record in decision-making under uncertainty, most famously powering AlphaGo's superhuman gameplay (Silver et al., 2017). More recently, MCTS has been adapted for LLM-guided planning, which frames LLM reasoning as a search problem over possible action sequences (Hao et al., 2023; Zhao et al., 2023; Gao et al., 2024; Lehnert et al., 2024). Building on this foundation, we propose Cost-Augmented MCTS, a novel extension where each action carries an explicit cost, and the planner must balance task completion against budget constraints. Different from traditional MCTS, which treats all actions as equally viable, our method prioritizes low-cost, high-reward paths, which aligns with real-world situations.

Our Cost-Augmented MCTS algorithm (**CATS**) incorporates cost-awareness at every step: actions are weighted by expense, and task constraints (e.g., "complete within $100") prune unfeasible paths. Tight constraints force the planner to quickly recognize impossible solutions, while loose ones encourage optimization for minimal cost. We benchmark leading LLMs, including GPT-4.1, Claude-3.7-Sonnet, and DeepSeek-R1, against our planner, analyzing their strengths in cost-sensitive scenarios.

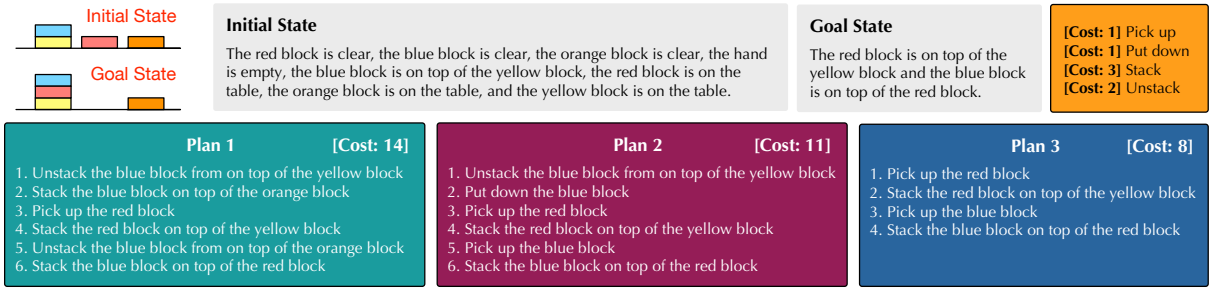

Figure 1: Blocksworld includes four actions, *pick up, put down, stack, and unstack*, each with an associated cost. Plans that are optimal in terms of length may not be cost-optimal. E.g., two plans with the same number of steps (6) can have different total costs (Plan 1 costs 14, while Plan 2 costs 11). Raw LLM planners can produce infeasible results, such as Plan 3.

Algorithm 1 Cost-Augmented MCTS

Input: Initial state s_0, state transition probability function p_θ, reward function r_θ, action generator p_ϕ, number of generated actions d, depth limit L, number of roll-outs N, and exploration weight w, Time Constraint T_0

Initialize memory of actions $A : \mathcal{S} \times T \mapsto \mathcal{A}$, children $c : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ and rewards $r : \mathcal{S} \times \mathcal{A} \times T \mapsto \mathbb{R}$
Initialize the state-action value function $Q : \mathcal{S} \times \mathcal{A} \times T \mapsto \mathbb{R}$ and visit counter $N : \mathcal{S} \mapsto \mathbb{N}$
Initialize time cost for each action: $T_a : \mathcal{A} \mapsto \mathbb{T}$

for $n \leftarrow 0, \ldots, N-1$ **do**
  $t \leftarrow 0$
  $T = T_0$
  while $N(s_t) > 0$ **do** ▶ Selection
    $N(s_t) \leftarrow N(s_t) + 1$
    $v(s_t, a) \leftarrow Q(s_t, a, T) + w\sqrt{\frac{\ln N(s_t)}{N(c(s_t,a))}}$
    $a_t \leftarrow \arg\max_{a \in A(s_t)} [v(s_t, a)]$
    $r_t = r(s_t, a_t, T)$, $s_{t+1} \leftarrow c(s_t, a_t)$
    $t \leftarrow t + 1$
    $T = T - T_a(a_t)$
  while s_t is not a terminal state $\wedge\ t \leq L \wedge T \geq 0$ **do**
    for $i \leftarrow 1, \ldots, d$ **do** ▶ Expansion
      Update $A(s_t, T) \leftarrow \{a_t^{(i)}\}_{i=1}^{d}$
      Update $c(s_t, a_t^{(i)}) \leftarrow s_{t+1}^{(i)}$
      Update $r(s_t, a_t, T) \leftarrow r_t^{(i)}$
    ▶ Simulation
    $a_{t+1} \leftarrow \arg\max_{a \in A(s_t)} r(s_t, a_t, T)$
    $r_t \leftarrow r(s_t, a_t, T)$, $s_{t+1} \leftarrow c(s_t, a_t)$
    $t \leftarrow t + 1$
  for $t' \leftarrow t, \ldots, 0$ **do** ▶ Back propagation
    Update $Q(s_{t'}, a_{t'}, T_{t'})$ with $\{r_{t'}, r_{t'+1}, \ldots, r_t\}$

Our results reveal key gaps in pure LLM planning and demonstrate how hybrid methods may outperform raw reasoning. Our study aims to push the boundaries of practical AI planning by combining LLMs with cost-aware search.

2 Cost-Augmented MCTS

Our approach leverages Monte Carlo Tree Search (MCTS) to enable Large Language Models (LLMs) to perform strategic planning. MCTS is a powerful algorithm that balances exploration and exploitation, iteratively building a reasoning tree where nodes represent states and edges represent actions. By combining the LLM's world model and reward function, MCTS efficiently searches for high-reward reasoning traces. The process consists of four key steps: Selection, Expansion, Simulation, and Back-propagation, which we detail below.

Selection starts from the root node (initial state) and navigates the tree to pick the most promising path for expansion. At each step, the algorithm chooses the next action using the Upper Confidence Bound for Trees (UCT) criterion, which balances exploiting high-value nodes (via the Q-function) and exploring less-visited nodes (via a visitation-based uncertainty term). This ensures that the search doesn't prematurely fixate on suboptimal paths. The selected path terminates at a leaf node, which may either be unexplored or a terminal state (e.g., a final answer in reasoning tasks).

Expansion grows the tree by adding new child nodes to the selected leaf. If the leaf isn't terminal, the LLM acts as an agent to propose d candidate actions (e.g., subquestions for math problems) and as a world model to predict their resulting states. These new nodes represent potential future reasoning steps. However, if the leaf is already terminal (e.g., a complete answer), expansion is skipped, and the algorithm proceeds to back-propagation. This step dynamically broadens the search space while respecting task boundaries.

Simulation estimates the expected future rewards (Q-values) by rolling out trajectories from the expanded node. Using a lightweight version of the reward function (for efficiency), the LLM simulates actions until reaching a terminal state. Unlike the expansion phase, which explores diverse ac-

		(Easy) ← **Success Rate** → (Hard)						(Easy) ← **Optimality** → (Hard)					
	PLAN LENGTH	$L = 2$	4	6	8	10	12	$L = 2$	4	6	8	10	12
TIGHT	DeepSeek-R1	0.62	0.15	0.02	0	0	0	1.00	1.00	1.00	n/a	n/a	n/a
	GPT-4.1	0.57	0.14	0.02	0	0	0	1.00	1.00	1.00	n/a	n/a	n/a
	Claude-3.7	0.37	0.09	0.02	0	0	0	1.00	1.00	1.00	n/a	n/a	n/a
	CATS w/ GPT-4.1	0.95	**0.91**	0.71	0.43	0.18	**0.06**	1.00	1.00	1.00	1.00	1.00	1.00
	CATS w/ Claude-3.7	**1.00**	0.83	**0.73**	**0.47**	**0.21**	0.04	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
LOOSE	DeepSeek R1	0.75	0.75	0.38	0.15	0	0	0.93	0.90	0.92	0.93	n/a	n/a
	GPT-4.1	0.53	0.44	0.18	0.06	0	0	1.00	0.92	0.92	0.93	n/a	n/a
	Claude-3.7	0.62	0.50	0.27	0.09	0	0	0.92	0.89	0.92	0.93	n/a	n/a
	CATS w/ GPT-4.1	**1.00**	**0.95**	**0.72**	0.43	0.14	**0.04**	1.00	0.95	0.95	**0.97**	**0.99**	**0.88**
	CATS w/ Claude-3.7	**1.00**	0.77	0.59	**0.49**	**0.32**	**0.04**	**1.00**	**1.00**	**0.98**	0.96	0.92	0.93

Table 1: We evaluate the planners' performance using two key metrics: **Success Rate** and **Optimality**.

tions, simulation prioritizes locally high-reward steps to approximate the path's quality. This trade-off reduces computational cost while still providing meaningful reward signals for back-propagation.

Backpropagation updates the Q-values along the traversed path using the rewards collected during simulation. This ensures promising actions receive higher values, guiding future selections. After a fixed number of iterations, the algorithm terminates and selects the best reasoning trace, either the highest-reward path or the most frequently visited one. In practice, we find that prioritizing the highest-reward path yields the most reliable results, as it directly optimizes for task performance.

Rewards. We build on the RAP framework (Hao et al., 2023) by introducing a cost-aware reward design. This reward estimates how likely the planner is to choose each next action, given the current context, as predicted by `GPT-4.1` or `Claude-3.7`. However, action log-probabilities aren't available for proprietary LLMs. Instead, we prompt the LLM to provide verbal likelihoods (e.g., "*Very Likely*") and map them to corresponding probabilities (e.g., *Impossible* = 0.01; *Very Unlikely*, *Unlikely*, *Somewhat Likely* = 0.50; *Likely*, *Very Likely*, *Almost Certain* = 1.00). To account for task-specific heuristics, we modify the goal reward to include a cost penalty, encouraging the planner to favor more efficient, lower-cost solutions. Specifically, the goal reward decreases linearly with the cost incurred so far (i.e., Goal Reward = 100 - Cost). This approach not only generalizes across LLMs but also optimizes for cost-aware planning, which is critical for real-world applications.

3 Data

We evaluate our approach on the BlocksWorld dataset (Valmeekam et al., 2022), a widely used benchmark for planning tasks (Bohnet et al., 2024). To systematically assess performance, we categorize test cases by difficulty, defined by the minimum steps (plan length) required to reach the goal state. Our dataset includes 45 step-2, 84 step-4, 152 step-6, 151 step-8, 112 step-10, and 46 step-12 scenarios, each involving up to five blocks. This setup allows us to measure how well different methods scale with increasing complexity.

World Model. We use PDDLGym (Silver and Chitnis, 2020), a Python framework with a Gym-compatible interface for PDDL-based planning, as our world model instead of relying on the LLM directly, since it offers faster and more reliable state updates for our benchmark tasks. While classic planners like Fast Downward (Helmert, 2006) guarantee precise solutions, they depend on handcrafted domain specifications. Recent hybrid approaches bridge this gap by combining LLMs' broad reasoning with classical planning, either by translating natural language into formal representations or by refining initial plans (Liu et al., 2023; Dagan et al., 2023; Guan et al., 2023). PDDLGym lets us harness the efficiency of symbolic planning while staying compatible with LLM-based exploration.

Cost. Actions can have different costs. As shown in Figure 1, the Blocksworld dataset includes four actions, *pick up, put down, stack, and unstack*, each with an associated cost (e.g., *stack* costs 3, while *pick up* costs 1). Under this setting, traditional plans that are optimal in terms of length may not be cost-optimal. For example, two plans with the same number of steps (6) can have different total costs (Plan 1 costs 14, while Plan 2 costs 11). To account for cost constraints, we recompute ground-truth optimal plans using exhaustive search. We also show that raw LLM planners can produce infeasible results, such as Plan 3 in Figure 1, emphasizing the need for structured search.

		(Easy) ← **Success Rate** → (Hard)						(Easy) ← **Optimality** → (Hard)					
	PLAN LENGTH	$L = 2$	4	6	8	10	12	$L = 2$	4	6	8	10	12
NO CONST.	DeepSeek-R1	**1.00**	**0.97**	**0.84**	**0.96**	**0.97**	**0.95**	0.88	0.87	0.88	0.86	0.86	0.86
	Claude-3.7	0.95	0.96	0.91	0.62	0.72	0.61	0.81	0.79	0.80	0.84	0.85	0.83
	GPT-4.1	0.75	0.38	0.40	0.34	0.39	0.36	0.88	0.85	0.80	0.84	0.88	0.86
	CATS w/ GPT-4.1	0.97	0.96	0.76	0.44	0.16	0.04	**0.99**	**1.00**	**0.98**	**0.98**	**0.95**	**0.95**

Table 2: Success rate and optimality with no task constraints. DeepSeek-R1 performs strongly as a raw planner.

		(Easy) ← **Efficiency** → (Hard)				
CATS	LENGTH	$L = 2$	4	6	8	10
GPT-4.1	TIGHT	3.06	1.42	2.37	**3.61**	3.45
	LOOSE	1.00	1.68	1.90	3.01	**4.35**
Claude-3.7	TIGHT	1.62	1.37	1.89	2.49	**3.54**
	LOOSE	2.15	1.51	1.80	2.33	**3.52**

Table 3: Efficiency of our CATS planner measured by the average number of iterations needed to find a valid solution, where fewer iterations mean faster convergence and lower computational cost.

4 Experiments

Metrics. We evaluate the planners' performance using two key metrics: **Success Rate** and **Optimality**. Success Rate checks whether the planner generates a valid plan, one that reaches the goal without violating task constraints. Optimality measures cost efficiency by comparing the generated plan's cost to the ground-truth optimal plan, with a ratio of 1.0 indicating perfect cost alignment. These metrics reveal not just whether a planner works, but how well it balances constraint satisfaction and cost-effectiveness.

Constraints. We compare our Cost-Augmented Tree Search (CATS) with raw LLM planners (GPT-4.1, Claude-3.7-Sonnet, DeepSeek-R1) under three budget conditions: TIGHT (task budget = optimal cost), LOOSE (budget = optimal + margin), and NO CONSTRAINT (unlimited budget). This design lets us examine how planners behave under strict task budgets versus more flexible scenarios where slight budget overruns may be acceptable.

Results. Tables 1 and 2 reveal several key insights: (a) Under tight (hard) constraints, raw LLMs struggle significantly. For example, success rates for `GPT-4.1` and `Claude-3.7` drop to 0% by step 8. In contrast, CATS maintains robust performance, achieving 71–73% success at step 6. This confirms that LLMs alone often fail under strict constraints, whereas CATS's cost-aware search reliably produces feasible plans. (b) In loose (soft) constraint settings, raw LLMs perform better. E.g., Claude-3.7 reaches 50% success at step 4 compared to just 9% under tight constraints. Still, CATS outperforms them, achieving near-perfect success on shorter plans (95% at step 4) and higher feasibility for longer ones. (c) In unconstrained settings, DeepSeek-R1 achieves high success rates but tend to lag in optimality, while CATS's success rate dips moderately on more complex tasks (longer plan lengths) as it prioritizes cost-optimal solutions.

Efficiency. We evaluate the efficiency of our CATS planner by measuring the average number of iterations needed to find a valid solution, where fewer iterations mean faster convergence and lower computational cost. Results are detailed in Table 3. Across all successful planning instances, CATS demonstrates strong performance. More complex planning tasks (longer plan lengths) require additional iterations, but the increase remains moderate. CATS paired with `Claude-3.7-Sonnet` outperforms the `GPT-4.1` variant, requiring fewer iterations to reach feasible plans, suggesting Claude's reward computations better guide the search.

Infeasibility Detection. When the task budget is set below the cost of the optimal plan, our CATS approach reliably identifies infeasibility, as the reward function classifies all possible next actions as "*Impossible*." Interestingly, we also found that raw LLM planners such as `GPT-4.1` perform surprisingly well. Their success rates in recognizing infeasible tasks remained high across different plan lengths: 100% for 2-step, 98% for 4-step, 95% for 6-step, and 73% for 12-step tasks. These results suggest that, advanced LLMs can effectively reason about and detect infeasibility, including in long-horizon scenarios.

5 Conclusion

We introduce CATS, a novel approach that brings cost-awareness to LLM-guided planning. While raw LLMs such as GPT-4.1 often struggle under tight budget constraints, CATS consistently generates cost-efficient plans by combining structured search with LLM reasoning. Our experiments show that CATS outperforms in both task success rate and cost optimality, especially in constrained settings where naive LLM planners tend to fail.

Limitations

While CATS demonstrates strong performance in cost-sensitive planning, there are limitations worth noting. MCTS requires multiple LLM calls per planning step, making it slower than raw LLM reasoning. We have mitigated this with efficient reward approximations, and real-time applications may need further optimizations. While our benchmark is representative, real-world planning involves fuzzy constraints (e.g., "comfortable travel budget"). Future work may integrate soft constraints or user feedback for adaptive budgeting.

References

Bernd Bohnet, Azade Nova, Aaron T Parisi, Kevin Swersky, Katayoon Goshvadi, Hanjun Dai, Dale Schuurmans, Noah Fiedel, and Hanie Sedghi. 2024. Exploring and benchmarking the planning capabilities of large language models. *Preprint*, arXiv:2406.13094.

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

Zitian Gao, Boye Niu, Xuzheng He, Haotian Xu, Hongzhang Liu, Aiwei Liu, Xuming Hu, and Lijie Wen. 2024. Interpretable contrastive monte carlo tree search reasoning. *Preprint*, arXiv:2410.01707.

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.

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. 2025. Large language models can solve real-world planning rigorously with formal verification tools. *Preprint*, arXiv:2404.11891.

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

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

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

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, Caiming Xiong, and Shafiq Joty. 2025. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *Preprint*, arXiv:2504.09037.

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*.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *Preprint*, arXiv:1712.01815.

Tom Silver and Rohan Chitnis. 2020. Pddlgym: Gym environments from pddl problems. In *International Conference on Automated Planning and Scheduling (ICAPS) PRL Workshop*.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025. Plangenllms: A modern survey of llm planning capabilities. *Preprint*, arXiv:2502.11221.

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. *Preprint*, arXiv:2402.01622.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large language models as commonsense knowledge for large-scale task planning. *Preprint*, arXiv:2305.14078.

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