Unlocking the Planning Capabilities of Large Language Models with Maximum Diversity Fine-tuning

Wenjun Li, Changyu Chen, Pradeep Varakantham
Singapore Management University
{wjli.2020, cychen.2020, pradeepv}@smu.edu.sg

Abstract

Large language models (LLMs) have demonstrated impressive task-solving capabilities through prompting techniques and system designs, including solving planning tasks (e.g., math proofs, basic travel planning) when sufficient data is available online and used during pre-training. However, for planning tasks with limited prior data (e.g., blocks world, advanced travel planning), the performance of LLMs, including proprietary models like GPT and Gemini, is poor. This paper investigates the impact of fine-tuning on the planning capabilities of LLMs, revealing that LLMs can achieve strong performance in planning through substantial (tens of thousands of specific examples) fine-tuning. Yet, this process incurs high economic, time, and computational costs for each planning problem variation. To address this, we propose Clustering-Based Maximum Diversity Sampling (CMDS), which selects diverse and representative data to enhance sample efficiency and the model's generalization capability. Extensive evaluations demonstrate that CMDS-l, a baseline method combining CMDS with language embeddings, outperforms random sampling. Furthermore, we introduce a novel algorithm, CMDS-g, which encodes planning task instances with their graph representations into the embedding space. Empirical results show that CMDS-g consistently outperforms baseline methods across various scales and multiple benchmark domains.

1 Introduction

System 1 competencies are characterized by fast, instinctive, and emotional responses, while System 2 competencies involve slower, more deliberate, and logical thinking processes (Kahneman, 2011). Prior studies (Valmeekam et al., 2024b; Pallagani et al., 2023; Liu et al., 2023; Guan et al., 2023; Kambhampati et al., 2024) have argued that LLMs struggle to generate valid plans in the automated planning domain due to weak System 2 competencies, despite demonstrating impressive planning capabilities in tasks such as Minecraft (Wang et al., 2024; Yuan et al., 2023) and household planning (Huang et al., 2022; Yao et al., 2022), where their System 1 competencies are more relevant (due to availability of large datasets online). Our study demonstrates that LLMs can indeed achieve System 2 competencies through fine-tuning with sufficiently large datasets.

This paper focuses on enhancing the planning capabilities of LLMs in rigorous, automated settings—termed System 2 planning. We distinguish this from System 1 planning, which covers tasks in environments like Minecraft and household activities. There are two key differences between these types of planning: 1) Complexity: System 2 planning is significantly more complex than System 1 planning, requiring deeper reasoning and more steps to solve; 2) Data availability: The amount of pre-training data available for System 1 tasks far exceeds that for System 2 tasks. For instance, the online data available for completing household tasks is much greater than that for finding optimal solutions to complex logistics problems.

Due to these challenges, LLMs initially exhibit weaknesses in automated planning (System 2) and researchers have been focusing on designing prompts and pipelines. However, these techniques are proven to have little improvement in improving LLM's capabilities in automated planning (Valmeekam et al., 2024b,a). A detailed related work section is provided in Appendix A. In this paper, we employ fine-tuning as a targeted approach and demonstrate that fine-tuning with sufficiently large datasets enables LLMs to achieve robust planning capabilities. Moreover, we provide an in-depth analysis of how different factors, such as data scaling, diversity, and complexity, impact the fine-tuning outcomes. We specifically examine how these factors influence both the planning capa-

Findings of the Association for Computational Linguistics: NAACL 2025, pages 3318–3340
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

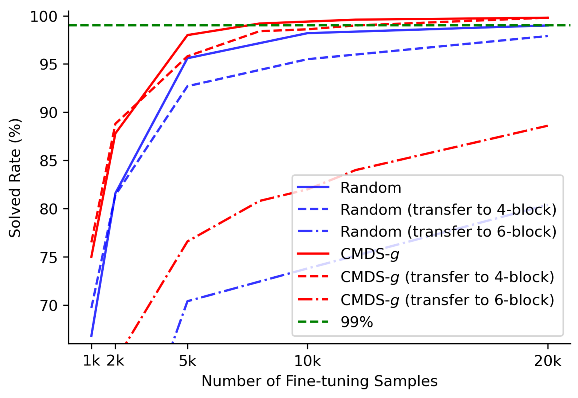


Figure 1: Scaling effect of fine-tuning `Llama-3-8b` in Blocksworld. Solved rate on hold-out (5-block) and transfer (4-block and 6-block) testing. A clearer comparison is in Appendix B.2.

bilities and generalization performance of LLMs. By addressing the gap in the current research, our study offers valuable insights into how to fine-tune LLMs more effectively and efficiently to enhance their planning competencies.

Fine-tuning both closed-source models like OpenAI's GPT-x (Brown et al., 2020) and open-source models such as Meta's Llama-x (Touvron et al., 2023) entails substantial time, economic, and computational costs. For instance, fine-tuning `GPT-3.5-turbo` on just 10,000 instances in Blocksworld (domain details provided in Experiment Setup) costs approximately 1,200 USD. Similarly, fine-tuning `Llama-3-8b` on this amount of data requires high-end GPUs and can take hundreds of GPU hours. Given these constraints, sample efficiency in fine-tuning is particularly crucial, especially for applications where models need frequent updates or rapid adaptation to new domains and complex tasks that require huge amounts of fine-tuning data. For example, housekeeping robots must efficiently adapt to various home layouts and furniture arrangements.

This paper aims to enhance the planning capabilities of LLMs while maximizing the sample efficiency and the model's generalization capability. We introduce a simple yet effective approach called Clustering-Based Maximum Diversity Sampling (CMDS), which selects diverse and representative data in the embedding space. Moreover, we propose CMDS-g to encode planning instances with their graph representations into the embedding space. A motivating example is illustrated in Figure 1. We generated tens of thousands of 5-block instances in Blocksworld and applied both Random sampling and CMDS-g to select fine-tuning samples. CMDS-g consistently outperforms Random at the same scale and demonstrated much higher sample efficiency as the data size increased. To achieve a 99% solved rate, Random sampling requires around 20,000 samples, whereas CMDS-g needs only about 7,500, effectively reducing the training time and computational costs by more than half (from 40 to 15 A100 GPU hours in our experiment). More importantly, CMDS-g exhibits superior generalization capabilities when the fine-tuned model is transferred to 4-block and 6-block test instances. These advantages become even more critical for large-scale planning tasks, where the fine-tuning data required to achieve satisfactory performance grows exponentially with task complexity. For example, exponentially more samples are needed for LLMs to perform equivalently on 6-block tasks as on 5-block tasks.

In summary, our research offers three key contributions. First, we provide a comprehensive analysis of how data scaling, diversity, and task complexity influence the fine-tuning outcomes of LLMs, particularly in automated planning. Our findings show that LLMs can achieve System 2 competencies through fine-tuning with sufficiently large datasets. Second, we introduce CMDS, a simple yet effective method for selecting diverse and representative data in the embedding space, thereby enhancing the sample efficiency of fine-tuning. Third, we propose CMDS-g, a novel approach that leverages graph representations to encode planning instances. Extensive experiments demonstrate that CMDS-g significantly boosts sample efficiency and generalization, consistently outperforming existing baseline methods.

2 Clustering-Based Maximum Diversity Sampling

Fine-tuning data consists of two components: queries and the associated responses. Previous studies (Zhou et al., 2024; Zhao et al., 2024) have shown that higher quality responses and more diverse queries lead to better LLM performance after fine-tuning. In automated planning, we can achieve the highest quality in responses by employing traditional planning solvers (e.g., Fast Downward (Helmert, 2006)) to find the optimal plans. As a result, how to capture the diversity of queries in fine-tuning data becomes the main challenge.

Unlike previous work, which relied on human

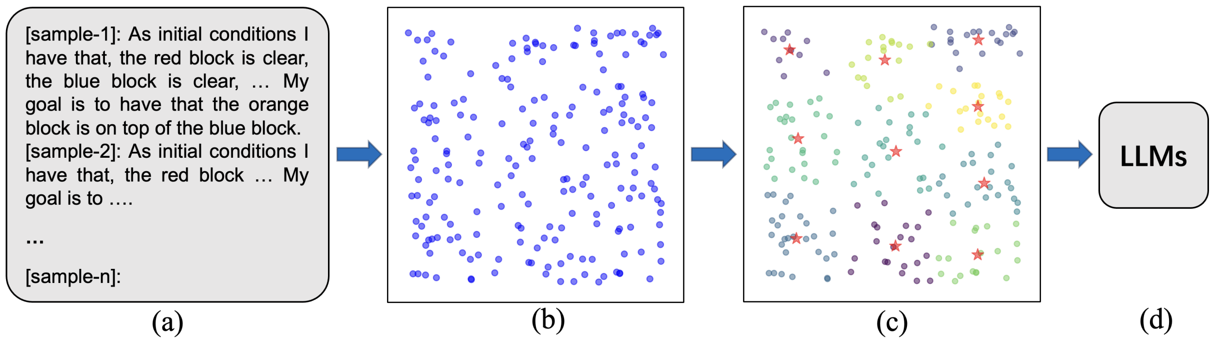

[sample-1]: As initial conditions I have that, the red block is clear, the blue block is clear, … My goal is to have that the orange block is on top of the blue block.
[sample-2]: As initial conditions I have that, the red block … My goal is to ….
...
[sample-n]:

Figure 2: Overview of CMDS. (a) Training samples in natural language. (b) Samples encoded in embedding space. (c) A subset of samples selected by CMDS. The red stars are the selected ones. (d) Fine-tune LLMs with the subset of samples.

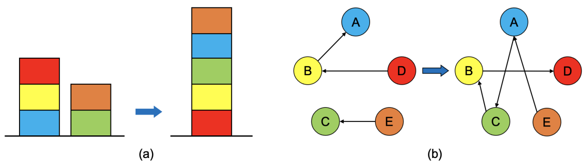


Figure 3: (a) An example task from Blocksworld. Left: initial configuration. Right: goal configuration. (b) A graph representation of the example task, where nodes are objects and edges are predicates.

resources to capture diversity in queries, we aim to automatically identify diverse and effective samples to achieve higher sample efficiency. The task of maximizing diversity within a subset of samples is known as the Maximum Diversity Problem (MDP, (Ghosh, 1996; Martí et al., 2013)), a well-studied NP-hard problem. This means that finding the exact optimal solution is computationally challenging for large datasets. In this paper, we introduce a simple yet effective method called Clustering-Based Maximum Diversity Sampling (CMDS) to identify representative samples, thereby ensuring diversity within the dataset.

2.1 Representing Planning Tasks as Graphs

MDP algorithms operate in numeric vector spaces, so the first step is to convert planning tasks from natural language into the embedding space.

Natural language can be efficiently encoded into the embedding space using tools such as Universal Sentence Encoder (Cer et al., 2018) and Sentence-BERT (Reimers, 2019). However, due to the narrative nature of planning tasks, tasks with distinct solutions can only differ in a few words in the natural language descriptions and result in highly similar language embeddings. For example, in Figure 3(a), switching the positions of the orange and green blocks in the initial configuration while keeping other conditions fixed changes the task descriptions slightly. The original task description is, “As initial conditions, I have that the orange block is clear, the orange block is on top of the green block, ...", whereas the altered task description is, “As initial conditions, I have that the green block is clear, the green block is on top of the orange block, ...". Despite having different solutions, these two tasks produce very similar language embeddings that are close in vector space. This issue makes subsequent diversity maximization ineffective. Additional results and analyses on the limitations of language embeddings are presented in Appendix B.7.

Therefore, a more representative embedding approach specifically tailored for planning tasks is necessary to capture the nuances between tasks. Inspired by (Rivlin et al., 2020; Silver et al., 2021), we use graph representations to encode planning tasks. Each planning task consists of an initial configuration and a goal configuration of objects. To solve the task, LLMs need to perform sequential actions to transform the initial configuration into the goal configuration. In the graph representation, each object is a node, and each predicate is a (directed) edge between two nodes. An example task from Blocksworld and its graph representation is shown in Figure 3. Once the planning tasks are converted into graphs, we can fully capture the task information and efficiently encode these graphs into vector embeddings by concatenating the vectors of the initial configuration and the goal configuration into a single graph representation. Details are provided in Appendix C.

2.2 Selecting Maximum Diversity Data

To enhance the effectiveness of downstream fine-tuning, we aim to select a subset of samples that

Algorithm 1: CMDS

```
Input: $N$ tasks, subset size $k$, embedding
       method $f(\cdot)$, distance metric $c(\cdot,\cdot)$
1  Initialize empty subset $\mathcal{D}$, embeddings set $\mathbb{E}$
2  for $i = 1$ to $N$ do
3      Encode task $t_i$ as vector embedding,
         $e_i = f(t_i)$
4      Insert $e_i$ into $\mathbb{E}$
5  end
6  Dimension reduction in embedding space
7  Perform clustering with distance metric
     $c(\cdot,\cdot)$ over the embeddings with $k$ clusters
8  Obtain $k$ centroids of the clusters
9  for $i = 1$ to $k$ do
10     In each cluster, find the $e_i$ closest to the
         cluster centroid
11     Retrieve the task instance associated
         with $e_i$ and add it to $\mathcal{D}$
12 end
Output: A subset of tasks, $\mathcal{D}$
```

are not only diverse but also representative of the overall dataset. We introduce a simple yet effective method called Clustering-Based Maximum Diversity Sampling (CMDS), designed to achieve this balance by selecting diverse and representative samples in the embedding space.

Assume we have a total number of N tasks and we want to find a subset of k tasks that maximize the overall diversity, defined as the sum of all pairwise distances between the k samples. To make the clustering more effective, we first perform dimension reduction (e.g., t-SNE (Van der Maaten and Hinton, 2008)) in the embedding space. Subsequently, we perform k-means clustering on the entire set of data points, with k used as the number of clusters. For each cluster, the data point closest to the cluster centroid is selected to represent the cluster in the final subset. When CMDS is applied with graph embeddings, we refer to it as CMDS-g.

Additionally, we propose an improved baseline method, CMDS-l, which combines CMDS with language embeddings. While language embeddings have been widely used to estimate dataset diversity, they have not previously been integrated with an automatic sample selection algorithm. Our extensive experiments demonstrate that CMDS-l outperforms random sampling. Due to its general applicability, CMDS-l is valuable for fine-tuning LLMs in domains beyond automated planning.

Our clustering-based method is compatible with any embedding approaches that can convert planning tasks from natural language to vector representations. The effectiveness of clustering depends on an appropriate distance metric; for example, we used the L2 norm distance for language embeddings and edit distance for graph embeddings in our experiments. The complete procedures of CMDS-l and CMDS-g are detailed in Algorithm 1.

3 Experiment Setup

Benchmark Domains: We utilize two widely adopted benchmark domains for automated planning from the International Planning Competition (IPC, (Long and Fox, 2003)). Detailed information about domain properties, actions, predicates, dataset generation, prompts, and other specifics can be found in Appendix C.

Blocksworld: This domain consists of blocks, a table, and a robot hand. Blocks can be placed on other blocks or on the table. A block with nothing on top is clear, and the robot hand can hold one block or be empty. The robot can perform four actions: pick-up, put-down, stack, and unstack. The goal is to transition from the initial to the goal configuration of blocks, uniquely identified by their colors. Task complexity mainly varies with the number of blocks. We created random instances with different block counts, separating training and testing tasks to ensure LLMs do not see testing tasks during fine-tuning. Detailed experimental settings are provided with each table and figure.

Logistics: The goal is to transport packages to specified locations. Locations are grouped by cities, with trucks moving packages within a city and airplanes moving packages between cities. Each city has one truck and an airport, and there can be multiple airplanes and packages. The Logistics domain is much more complex than Blocksworld, requiring longer plans. We generated distinct instances with varying numbers of cities, locations, packages, and airplanes.

Base Models: We use `GPT-3.5-turbo-0125` and `Llama-3-8b` for Research Question 1 (RQ-1), and include `Llama-2-7b` (Touvron et al., 2023) for Research Question 2 (RQ-2). `GPT-4` (Achiam et al., 2023) was omitted from our experiments due to its limited experimental access and lack of public availability for fine-tuning.

Baselines: We compare CMDS-g with two baselines: Random and CMDS-l. Random uniformly samples k instances from the entire training dataset. CMDS-l differs from CMDS-g only in the embedding method and distance metric, as described in section 2 and Algorithm 1.

Evaluation: In our experiments, LLMs respond in natural language, and the generated plans are translated to PDDL and then verified using VAL (Howey et al., 2004). A plan is considered correct as long as it successfully transitions from the initial configuration to the goal configuration. An analysis of the optimality rate of the generated plans is provided in Appendix B.5.

Prompt: We use two types of prompts: one-shot and zero-shot. The one-shot prompt includes domain instructions, an example task with its solution, and a query task. The zero-shot prompt contains only domain instructions and the query task. Further details about the prompt settings are available in Appendix C. We found that fine-tuned models perform better with zero-shot prompts because they have seen many examples during fine-tuning, enabling accurate and concise responses. Conversely, un-fine-tuned models perform better with one-shot prompts. Therefore, we evaluate un-fine-tuned models with one-shot prompts and fine-tuned models with zero-shot prompts.

4 Experiment Results

4.1 RQ-1. How does fine-tuning impact the planning capabilities of LLMs?

To evaluate the planning capabilities and transferability of LLMs precisely, we prepared multiple testing tasks with varying numbers of blocks in Blocksworld. Specifically, there are 100, 500, 500, and 500 testing tasks for 3-block, 4-block, 5-block, and 6-block settings, respectively. Meanwhile, the Logistics domain has too many variables (e.g., the number of cities, locations, trucks, airplanes, etc.) that affect task complexity, making it challenging to identify which variable impacts task difficulty the most. Therefore, we randomly selected 300 instances with varying numbers of variables as the testing data for the Logistics domain.

4.1.1 Significant Improvement Achieved Through Fine-tuning

We first assessed the planning capabilities of LLMs prior to fine-tuning, with results shown in Table 1. Notably, LLMs without fine-tuning exhibit weak planning capabilities, and these results align with existing criticisms of LLMs' planning capabilities in the literature.

Next, we evaluated the planning capabilities of LLMs after fine-tuning them on planning tasks using two sample sizes: small (100 samples) and large (1000 samples). Table 1 presents the test performance of LLMs after fine-tuning. Remarkably, even with a small number of samples, fine-tuning significantly enhanced the LLMs' planning capabilities. Additionally, we tested the fine-tuned models on the dataset from (Valmeekam et al., 2024b), where `GPT-3.5-turbo` and `Llama-3-8b` achieved solved rates of 90.6% and 84.3%, respectively. Humans are reported in (Valmeekam et al., 2024b) to have a solved rate of 78%. These findings challenge the prevailing notion that LLMs are inherently weak at planning.

After closely investigating the failure cases of pre-fine-tuning models, we found that the failures were not due to poor instruction following or template mismatches, but because the LLMs could not provide correct plans. In contrast, fine-tuning even on a small amount of data significantly unlocks LLMs' planning capabilities. This suggests that LLMs' poor initial performance is due to a lack of exposure to automated planning tasks during pre-training.

We believe that high-level planning skills cannot be acquired through prompting techniques or pipeline designs alone. Models fine-tuned on planning domains can serve as the backbone for developing more advanced applications. This strategy is more effective than solely relying on prompting strategies or pipelines. This critical insight, often overlooked in the community, provides valuable guidance on how to effectively enhance LLMs' planning capabilities.

4.1.2 Data Scaling Effect

The empirical results of the data scaling effect in both Blocksworld and Logistics are provided in Appendix B.2 due to space limits. Overall, the scaling effect is similar to what we have shown in the motivation example. LLM's planning capabilities improve as the amount of fine-tuning data increases. This is because the responses in fine-tuning data are of optimal quality, regarding accuracy, efficiency, and validity. However, this improvement follows an asymptotic pattern – while initial increases in sample size lead to substantial performance gains, the rate of improvement (slope) diminishes as more samples are added. This suggests that exponentially increasing amounts of data are needed to further

Domain		GPT-3.5-turbo			Llama-3-8b		
		Pre-FT	Post-FT		Pre-FT	Post-FT	
			$k = 100$	$k = 1000$		$k = 100$	$k = 1000$
Blocksworld	3-block	8.0% (8/100)	65.0%	98.0%	0.0%	57.0%	85.0%
	4-block	3.6% (18/500)	61.8%	91.9%	0.0%	43.9%	91.3%
	5-block	1.2% (6/500)	36.4%	65.8%	0.0%	19.6%	46.4%
	6-block	0.4% (2/500)	15.4%	34.0%	0.0%	3.6%	15.0%
Logistics		0.7% (2/300)	7.5%	73.0%	0.0%	2.8%	72.7%

Table 1: Task solved rate of pre-fine-tuning (Pre-FT) and post-fine-tuning (Post-FT) models in Blocksworld and Logistics domain. k is the number of random samples used in fine-tuning. We use 4-block samples for fine-tuning in Blocksworld and samples with varying numbers of variables for fine-tuning in Logistics. GPT and Llama results are collected across three and five random seeds respectively.

enhance LLMs' planning capabilities.

4.1.3 Transferability

We also examined the transfer performance of LLMs across different tasks and domains after fine-tuning. Due to space constraints, detailed cross-domain transferability results are presented in Appendix B.3.

In-domain Transferability: We assessed LLM's transfer performance to different tasks within the same domain after fine-tuning. Specifically, `GPT-3.5-turbo` and `Llama-3-8b` were fine-tuned with 1,000 samples of 4-block, 5-block, and 6-block tasks, and then assessed on tasks with different block numbers than those used in fine-tuning. Note that we did not fine-tune the base models with 1,000 3-block samples because the 3-block setting does not provide enough distinct samples. The in-domain transferability results are shown in Table 2, and we make two key observations:

1. LLMs exhibit good in-domain transferability and can apply their acquired planning capabilities to new challenges. For example, GPT-3.5 fine-tuned with 5-block tasks transfers well (68.0%) to 6-block tasks, even when the tasks involve unseen block colors.

2. LLMs show better "strong-to-weak" transferability than "weak-to-strong" transferability. Specifically, LLMs transfer more effectively to less complex tasks. For instance, both GPT-3.5-6block and Llama-3-6block demonstrate better overall performance on tasks. This suggests that fine-tuning LLMs on more complex planning tasks can facilitate better transfer to simpler tasks.

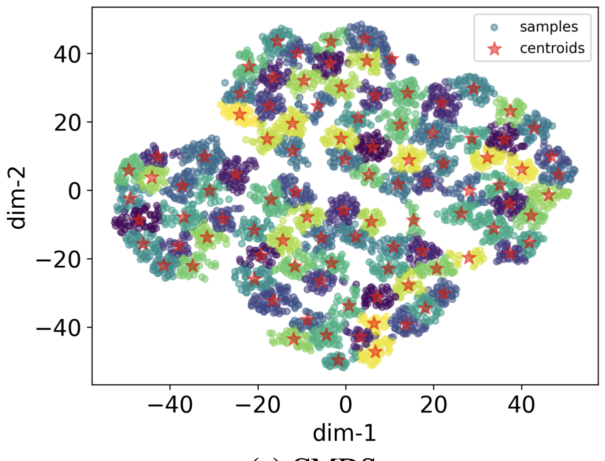
Model	3-block	4-block	5-block	6-block
GPT-4block	98.0%	91.9%	65.8%	34.0%
GPT-5block	81.0%	88.0%	84.8%	68.0%
GPT-6block	74.0%	75.4%	77.0%	70.6%
Llama-4block	85.0%	91.3%	46.4%	15.0%
Llama-5block	73.0%	82.3%	71.6%	39.2%
Llama-6block	63.0%	65.6%	69.2%	58.6%

Table 2: In-domain transferability of fine-tuned LLMs. Models fine-tuned on x-block tasks are denoted as GPT-$x$block and Llama-$x$block. Fine-tuned models are evaluated on x-block tasks within the Blocksworld domain.

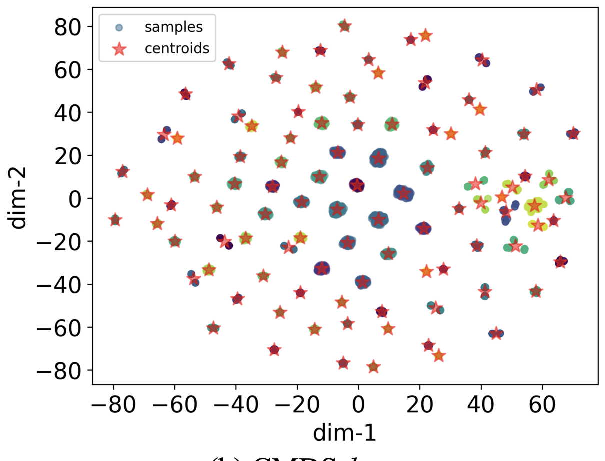
Cross-domain Transferability: While LLMs exhibit high in-domain transferability, our findings indicate their cross-domain transferability remains limited. We provide extensive results on this in Appendix B.3. Acquiring generalizable capabilities across different automated planning domains is particularly challenging for LLMs due to the substantial differences in actions, predicates, and objects. To address this, we fine-tuned LLMs on tasks from multiple domains collectively, aiming to develop multi-task capabilities. A detailed analysis of how data composition influences fine-tuning outcomes is included in the extended experimental section in Appendix B.4. In summary, although fine-tuning on mixed data effectively equips LLMs with multi-task planning capabilities, it introduces slightly higher variance in performance outcomes.

4.2 RQ-2: Can we identify the most effective samples to enhance sample efficiency?

In this section, we demonstrate that CMDS-g identifies the most representative and effective samples,



(a) CMDS-*g*



(b) CMDS-*l*

Figure 4: Subset samples selected by (a) CMDS-g and (b) CMDS-l. Each point represents a planning task in Blocksworld, encoded in the embedding space. The sample point closest to each cluster centroid is selected for the subset. Note that clusters in CMDS-l appear to have only a few dots as the points within each cluster are so closely packed together.

thereby enhancing sample efficiency during fine-tuning and improving the model's generalization capability afterward. Additionally, we found that CMDS-l also outperforms random sampling, highlighting the value of the CMDS approach. We simplified testing data composition for Blocksworld, as detailed performance on different types of tasks is unnecessary for analysis in this section. We uniformly sampled from the four types of testing tasks (i.e., 3-block, 4-block, 5-block, 6-block) to collectively create a testing dataset with 1000 instances. Testing data for Logistics remains consistent with the previous sections.

First, we illustrate the effectiveness of the clustering-based method in solving the maximum diversity problem. Figure 4 shows the subsets of samples identified by CMDS-g and CMDS-l. In both plots, the selected samples (red stars) are located nearest to the centroids of each cluster, ensuring a diverse and broad coverage of the entire vector space. Notably, the data points in each cluster for CMDS-l (Figure 4b) are very dense and closely packed, reflecting the limitations of language embeddings in capturing the complexity and structure of planning tasks. This results in clusters that are more influenced by narrative differences than by task structures, leading to a lack of diversity in the selected samples regarding task structure. In contrast, CMDS-g (Figure 4a) utilizes graph embeddings to represent tasks with greater expressiveness. This approach allows CMDS-g to capture the structural nuances of planning tasks more effectively, resulting in more representative clustering and a more diverse selection of samples.

Next, we empirically demonstrate the effectiveness of CMDS-g at different scales. In Blocksworld, we created five thousand instances with varying numbers of blocks as the training dataset. Using Random, CMDS-l, and CMDS-g, we selected task subsets of varying sizes ($k =$100, 200, 400, and 1,000), fine-tuned the base models, and then assessed their performance on the hold-out testing data. The training and testing data for Logistics are the same as in previous experiments. Comprehensive results for both Blocksworld and Logistics are provided in Tables 3.

Notably, CMDS-g consistently outperforms both Random and CMDS-l across different scales in both domains. CMDS-g significantly improves sample efficiency in fine-tuning, particularly at smaller scales. For instance, CMDS-g with $k =$ 100 achieves a 71.7% solved rate, a performance level that requires approximately 200 samples when using Random or CMDS-l. While the performance advantage of CMDS-g diminishes with larger sample sizes—reflecting the diminishing returns from additional samples—this is attributed to the nature of fine-tuning rather than a limitation of CMDS-g. Therefore, CMDS-g is particularly advantageous in scenarios where fine-tuning incurs high economic and computational costs, making it ideal for applications with limited resources or strict efficiency requirements. Furthermore, as highlighted in the motivation example, CMDS-g

Domain	Base Model	Algorithm	Subset Size $k = 100$	$k = 200$	$k = 400$	$k = 1000$
Blocksworld	GPT-3.5-turbo	Random	61.8±1.9%	72.4±1.8%	81.9±1.5%	91.9±1.2%
		CMDS-l	60.4±1.6%	73.2±1.4%	82.8±1.3%	92.5±1.0%
		CMDS-g	**71.7±1.7%**	**77.6±2.0%**	**85.8±1.3%**	**94.9±0.8%**
	Llama-2-7b	Random	28.8±3.6%	30.3±8.6%	50.0±1.0%	74.6±3.7%
		CMDS-l	24.8±4.4%	32.6±3.8%	51.5±1.4%	74.2±2.9%
		CMDS-g	**32.5±2.1%**	**37.2±4.5%**	**52.9±1.3%**	**75.7±0.7%**
	Llama-3-8b	Random	43.9±2.9%	54.2±6.8%	67.6±3.3%	86.3±0.8%
		CMDS-l	42.0±3.9%	53.6±5.4%	69.2±4.5%	87.1±0.7%
		CMDS-g	**51.2±5.3%**	**59.5±3.7%**	**71.6±2.4%**	**88.5±0.7%**
Logistics	GPT-3.5-turbo	Random	7.5±2.5%	20.6±2.1%	45.8±3.8%	62.3±2.3%
		CMDS-l	8.3±1.5%	19.3±2.7%	42.2±2.1%	66.3±1.7%
		CMDS-g	**11.1±1.6%**	**25.0±1.2%**	**51.8±1.2%**	**73.0±1.1%**
	Llama-2-7b	Random	0.4±0.1%	1.9±0.3%	11.9±1.3%	53.0±2.1%
		CMDS-l	**0.8±0.3%**	2.0±0.4%	12.6±3.0%	54.1±2.4%
		CMDS-g	0.4±0.2%	**4.7±0.9%**	**15.5±1.5%**	**56.1±1.8%**
	Llama-3-8b	Random	2.8±1.1%	10.3±0.7%	44.4±1.5%	72.7±3.1%
		CMDS-l	3.9±1.0%	8.8±1.5%	46.8±3.8%	73.2±1.8%
		CMDS-g	**5.3±1.0%**	**12.9±1.6%**	**49.8±2.6%**	**75.3±0.8%**

Table 3: Task solved rate in the Blocksworld and Logistics domain. Results are presented as mean ± standard deviation. GPT and Llama results are collected across three and five random seeds respectively.

remains valuable as task complexity increases, requiring significantly less data to converge and generalize, thereby reducing costs substantially.

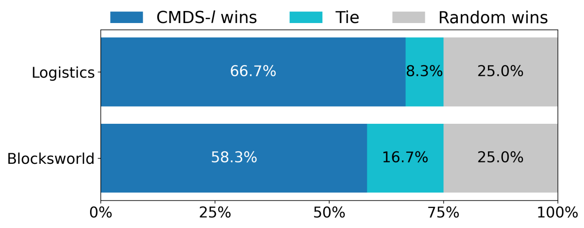


Figure 5: Performance comparison between CMDS-l and Random.

Finally, we present a detailed comparison of the performance between CMDS-l and Random. Figure 5 illustrates the win-tie-loss comparison between these methods in the Blocksworld and Logistics domains. CMDS-g is excluded from this comparison because it consistently outperforms both. The results indicate that CMDS-l achieves more wins across both domains, further validating the effectiveness of the CMDS approach. Additionally, due to its broad applicability of language embeddings, CMDS-l is valuable for fine-tuning LLMs in domains beyond automated planning.

5 Conclusion

In this paper, we extensively study how data scaling, diversity, and task complexity affect fine-tuning outcomes in automated planning domains. Our findings show that LLMs can exhibit strong planning capabilities through fine-tuning with sufficiently large datasets, challenging the notion that LLMs are inherently weak in System 2 planning. To enhance sample efficiency and reduce fine-tuning costs, we introduce the Clustering-Based Maximum Diversity Sampling (CMDS) approach, which ensures a broad and representative sample set. Our methods, CMDS-g (using graph embeddings) and CMDS-l (using language embeddings), consistently outperform random sampling. Notably, CMDS-g significantly enhances both the sample efficiency in fine-tuning and the generalization performance of the fine-tuned models. This research provides a thorough investigation of LLMs' planning capabilities and offers insights into effectively and efficiently developing these capabilities.

6 Limitations

While the proposed algorithm enhances LLMs' performance in automated planning with higher sample efficiency and better generalization performance, the planning capabilities acquired in one domain can hardly transfer to new domains. Although the mixed-data fine-tuning approach proves effective, it also introduces slightly higher variance in outcomes, indicating the need for further refinement and stability enhancements. Addressing these limitations will be the focus of our future work. Cross-domain transferability in planning remains an underexplored area, highlighting a significant opportunity for further investigation.

Acknowledgments

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-017).

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*.

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. 2024. Attnlrp: attention-aware layer-wise relevance propagation for transformers. *arXiv preprint arXiv:2402.05602*.

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 Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jay B Ghosh. 1996. Computational aspects of the maximum diversity problem. *Operations research letters*, 19(4):175–181.

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

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

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.

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

Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Allen Lane.

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*.

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*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Derek Long and Maria Fox. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59.

Rafael Martí, Micael Gallego, Abraham Duarte, and Eduardo G Pardo. 2013. Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics*, 19:591–615.

OpenAI. 2024. Introducing openai o1-preview. Accessed: 2024-9-12.

Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Biplav Srivastava, Lior Horesh, Francesco Fabiano, and Andrea Loreggia. 2023. Understanding the capabilities of large language models for automated planning. *arXiv preprint arXiv:2305.16151*.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Or Rivlin, Tamir Hazan, and Erez Karpas. 2020. Generalized planning with deep reinforcement learning. *arXiv preprint arXiv:2005.02305*.

Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2021. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11962–11971.

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. 2024a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2024b. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. 2024. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. *Advances in Neural Information Processing Systems*, 36.

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.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yue Yu, Rongzhi Zhang, Ran Xu, Jieyu Zhang, Jiaming Shen, and Chao Zhang. 2023. Cold-start data selection for better few-shot language model fine-tuning: A prompt-based uncertainty propagation approach. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2499–2521.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. 2023. Skill reinforcement learning and planning for open-world long-horizon tasks. *arXiv preprint arXiv:2303.16563*.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. *arXiv preprint arXiv:2402.04833*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Peide Zhu and Claudia Hauff. 2022. Unsupervised domain adaptation for question generation with domain data selection and self-training. In *2022 Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2388–2401. Association for Computational Linguistics (ACL).

A Related Work

Enhancing LLMs' Reasoning and Planning Capabilities with Prompting. Prompting involves guiding the input to a Large Language Model (LLM) using templates or cues to produce the desired output. Techniques like Chain of Thought (CoT, (Wei et al., 2022)) and Tree of Thoughts (ToT, (Yao et al., 2024)) have demonstrated the ability to enhance complex reasoning and planning by breaking down tasks into intermediate reasoning steps. These prompting methods have proven effective in solving mathematical problems (Huang et al., 2022; Yao et al., 2022) and household tasks (Wei et al., 2022; Yao et al., 2024). However, their effectiveness in automated planning domains has been limited, largely due to the inherent complexity of these tasks. For example, (Valmeekam et al., 2024b) showed that CoT prompting resulted in only a 1% improvement in Blocksworld, with GPT-4 (Achiam et al., 2023) achieving a 34.6% solved rate without CoT prompting and 35.6% with it.

LLMs with access to external tools. Due to LLMs' tendency to generate responses with hallucinations and their lack of strict adherence to actions and predicates, some approaches integrate external tools to provide feedback. For example, (Liu et al., 2023) and (Guan et al., 2023) utilize the Planning Domain Definition Language (PDDL) executor to verify the validity of generated plans and provide feedback (such as explanations for why plans are not executable) to aid LLMs in reflection and plan refinement. However, these methods do not fundamentally enhance the LLMs' inherent planning capabilities and result in complex integration engineering and potential high latency. Furthermore, frequent use of external tools can be costly and prone to reliability issues. For instance, the effectiveness of traditional planners depends heavily on the accuracy of translating natural language to PDDL; inaccuracies in this translation can lead to incorrect plans or failed validations.

Enhancing LLM's task-specific capabilities with fine-tuning. Fine-tuning refers to the process of updating the parameters of a pre-trained model using task-specific data. Initial studies, such as Less Is More for Alignment (LIMA, (Zhou et al., 2024)), have investigated the effects of training data quantity and quality on fine-tuning outcomes in typical natural language processing tasks like Q&A and creative writing. However, to the best of our knowledge, there has been no comprehensive study on how fine-tuning data (considering aspects like quantity, diversity, composition, etc.) affects the planning capabilities of LLMs. This paper pioneers the exploration of these research questions.

Large Reasoning Models. Recently, OpenAI released o1-preview (OpenAI, 2024), a model trained to generate an internal thought before answering questions using effective reinforcement learning with human feedback (RLHF) techniques. o1 pioneers the application of scaling laws during inference and showcases a notable improvement in reasoning and planning capabilities compared to existing LLMs. Our proposed method focuses on enhancing the base model during the supervised fine-tuning (SFT) stage, a critical step in preparing a strong initial model for RLHF. Given the proven effectiveness of CMDS in the SFT stage, we believe that combining CMDS with advanced RLHF techniques could further enhance the model's reasoning and planning performance and potentially reduce overall training costs. We consider this integration a potential direction for future work.

Data Selection. Existing work on data selection for training LLMs, such as (Yu et al., 2023) and (Zhu and Hauff, 2022), addresses related challenges but differs from ours in two key ways. First, our work focuses specifically on enhancing logical reasoning and planning capabilities in LLMs, a critical and increasingly important domain. Our data selection method has the potential to be applied to training large reasoning models like o1, whereas (Yu et al., 2023) centers on text classification and (Zhu and Hauff, 2022) targets question generation from text passages. Second, and more importantly, our approach differs fundamentally in methodology. Both (Yu et al., 2023) and (Zhu and Hauff, 2022) rely on natural language embeddings, similar to our baseline method, CMDS-l. However, as demonstrated in Section 4.2, language embedding-based methods struggle to effectively distinguish nuanced planning instances, highlighting the limitations of these approaches in our targeted domain.

B Extended Experiment Results

In this section, we present extended experimental results, covering various aspects of our proposed method and details about the planning capabilities of LLMs. These include its performance on imbalanced datasets (B.1), the detailed effects of dataset scaling (B.2), cross-domain transferability (B.3), the impact of data composition on fine-tuning outcomes (B.4), an analysis of the optimality of plans generated by LLMs (B.5), a detailed performance analysis (B.6), further investigation into the limitations of language embeddings (B.7), and a visualized evaluation via attention (B.8).

B.1 Imbalanced Dataset

The fine-tuning data in the previous experiments were collected through randomization with unique checking, ensuring a uniform and balanced training dataset. However, in real-world scenarios, fine-tuning data often varies in quality and distribution. To simulate this, we conducted additional experiments using imbalanced fine-tuning data. Since no standard metric exists to measure dataset imbalance in automated planning, we devised a heuristic imbalance coefficient to control the level of imbalance. Specifically, we randomly removed data points from p*100% of all clusters (identified by CMDS-g such that the samples represent diverse planning instances based on task structures) to create datasets with imbalanced distributions. Within the clusters selected for data removal, we retained $j \in [1, 5]$ points. As shown in Figure 6, this process resulted in areas with dense examples and others with sparse examples.

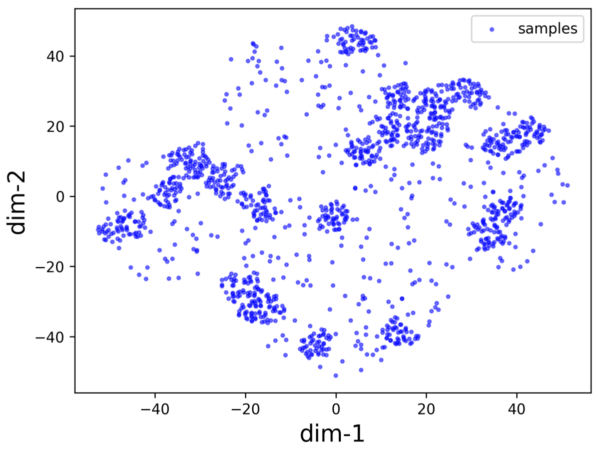


Figure 6: Imbalanced data distribution in the Blocksworld domain.

We created datasets with varying degrees of imbalance by adjusting the percentage of clusters removed from the training dataset. Note that this imbalance coefficient is an approximate metric designed to control the dataset's imbalance level, and it does not strictly scale linearly with the degree of imbalance.

We employed Random, CMDS-l, and CMDS-g sampling methods on the imbalanced datasets and conducted experiments using Llama-3-8b. The results, depicted in Figure 7, lead to two key observations: (1) CMDS-g consistently outperforms both Random and CMDS-l by a significant margin; (2) generally, CMDS-l outperforms Random, winning in 9 out of 11 imbalance coefficient scales. Additionally, the results shown in Figure 7 are worse compared to those obtained with a balanced dataset, indicating that a balanced dataset is beneficial for enhancing model capabilities.

B.2 Data Scaling Effect

In this section, we first assess the scaling effect on the model's in-domain transferability. Specifically, we evaluate models trained on 5-block tasks (as shown in the motivation example) using held-out 4-block and 6-block test tasks, with the results presented in Figure 8b. We make three key observations: (1) Transfer performance follows the same scaling effect, exhibiting an asymptotic pattern—initial increases in sample size result in significant performance gains, but the rate of improvement diminishes as more samples are added; (2) Transfer performance is generally worse than standard performance, i.e., test

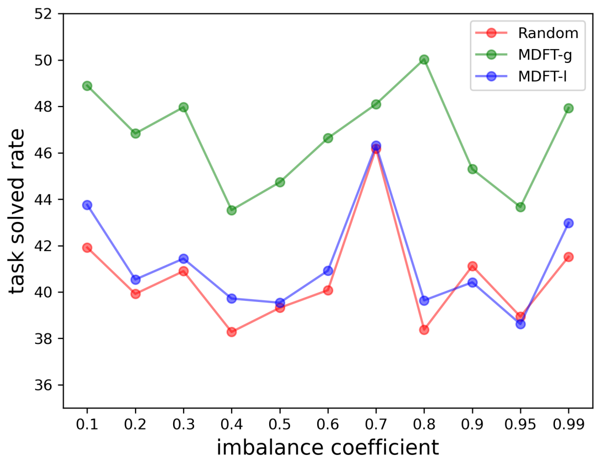


Figure 7: Fine-tuning performance vs imbalance coefficient when $k = 100$ in Blocksworld. Results are collected with Llama-3-8b and averaged across five independent runs.

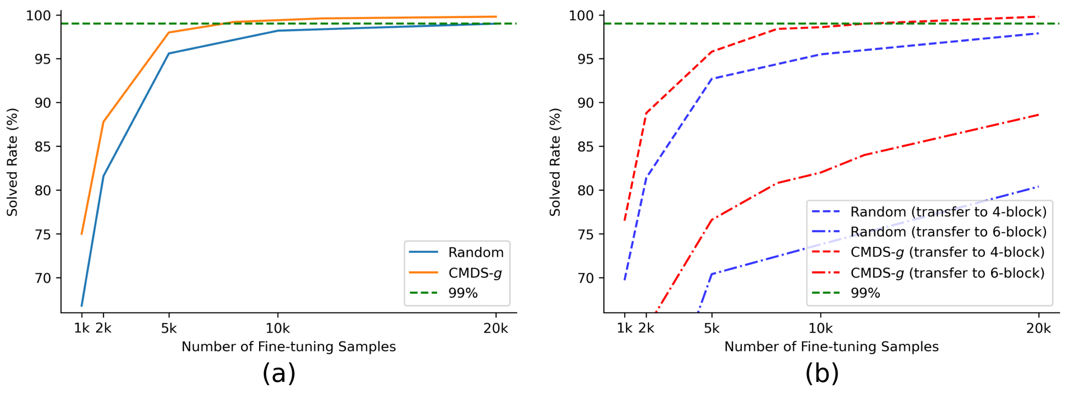


Figure 8: Scaling effect of fine-tuning Llama-3-8b in Blocksworld. (a) Model fine-tuned on 5-block samples and evaluated on hold-out 5-block test tasks. (b) Model fine-tuned on 5-block samples and evaluated on hold-out 4-block and 6-block test tasks.

performance on tasks that match the training settings; (3) CMDS-g achieves higher sample efficiency in transfer performance. For instance, CMDS-g attains a 99% solved rate with approximately 7,500 samples, whereas Random requires around 20,000 samples to reach this performance. More importantly, using fine-tuning data selected by CMDS-g enables the model to generalize better to different tasks. These advantages underscore the value of our proposed method, particularly as task complexity increases and the data requirements rise exponentially. Our approach significantly enhances sample efficiency and generalization capability while reducing the substantial time, economic, and computational costs involved.

Furthermore, we closely examine the impact of data scaling on the planning capabilities of LLMs using a smaller-scale experiment. We conducted tests in the 4-blocks setting of Blocksworld with GPT-3.5-turbo-0125 and Llama-3-8b. The results are summarized in Figure 9. Our findings indicate that GPT-3.5-turbo exhibits superior planning capabilities compared to the Llama models, both before and after fine-tuning. However, when fine-tuned with 4000 samples, Llama-3-8b outperforms GPT-3.5-turbo. This discrepancy is largely due to the limited number of fine-tuning epochs applied to GPT-3.5-turbo, driven by the high economic costs, while the Llama models were fine-tuned over significantly more epochs.

As observed consistently, LLMs' planning capabilities improve with increased fine-tuning data, due to the high quality of responses in terms of accuracy, efficiency, and validity. However, this improvement follows an asymptotic pattern, where the rate of performance gains diminishes as more samples are added. This indicates that substantial data is needed to fine-tune LLMs effectively for complex System 2 planning tasks, with data requirements growing exponentially with task complexity. In such cases, our proposed method can significantly reduce the time, economic, and computational costs associated with fine-tuning.

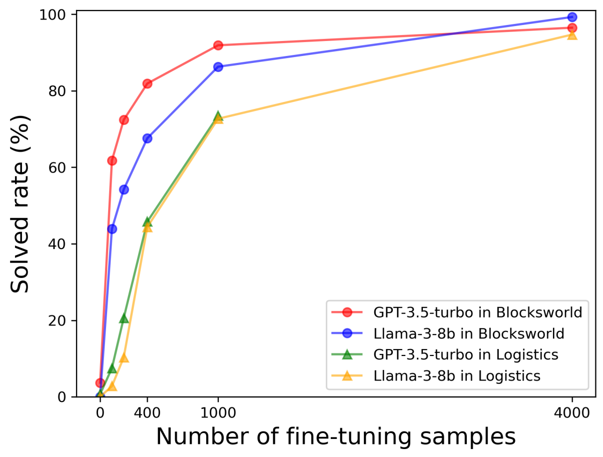


Figure 9: Model performance versus the number of fine-tuning samples. GPT-3.5-turbo finetuned with 4000 Logistics samples is omitted in this experiment due to high economic costs.

B.3 Cross-domain Transferability

To evaluate cross-domain transferability, we fine-tuned the base models in one domain and then tested their performance in another. Specifically, we fine-tuned the models on 200 tasks from the Blocksworld domain and evaluated their performance in the Logistics domain. Conversely, we fine-tuned the models on 400 tasks from the Logistics domain and assessed their performance in the Blocksworld domain. We intentionally used relatively small sample sizes in this experiment to avoid overfitting the LLMs to the specific response patterns and dynamics of a single domain. The results of these cross-domain transferability tests are presented in Table 4.

			Fine-tuning Tasks	
Test Domain	**Model**	**un-finetuned**	**Blocksworld**	**Logistics**
Blocksworld	GPT-3.5	3.7%	72.4%	3.1%
	Llama-3	0%	54.2%	0.3%
Logistics	GPT-3.5	0.7%	0.3%	45.8%
	Llama-3	0%	0%	44.4%

Table 4: LLM's transferability across domains.

The findings reveal that LLMs exhibit poor transferability across different planning domains after fine-tuning, in some cases performing worse than models that were not fine-tuned. This is due to the substantial differences between automated planning domains in terms of actions, predicates, and objects, which hinder the LLMs' ability to generalize across domains. To develop LLMs with effective multi-task capabilities, it is therefore necessary to fine-tune them on tasks drawn from multiple domains simultaneously. In the next section, we provide a detailed analysis of how data composition influences fine-tuning outcomes.

B.4 Data Composition

To assess the impact of data composition on fine-tuning, we created a mixed dataset by randomly sampling task instances from both the Blocksworld and Logistics domains. This mixed dataset was used to fine-tune GPT-3.5-turbo and Llama-3-8b models. The dataset was composed of an equal number of samples from each domain, which were randomly shuffled before fine-tuning. For example, a dataset with $k = 100$ includes 100 samples from Blocksworld and 100 samples from Logistics, making it directly comparable to a dataset of $k = 100$ pure samples from a single domain. The effects of data composition on fine-tuning outcomes are summarized in Table 5.

Our key finding is that fine-tuning on mixed-domain data does not degrade model performance; in fact, it slightly improves it. Statistically, mixed data outperforms pure data in 5 out of 7 settings in both the Blocksworld and Logistics domains. This result is significant because, despite the poor cross-domain transferability of LLMs, we can enhance their planning capabilities across different domains by fine-tuning on a combined dataset. This conclusion is both crucial and promising, indicating that LLMs can acquire strong cross-domain planning capabilities through fine-tuning on sufficiently large and diverse datasets from multiple domains. These findings underscore the potential of using mixed planning tasks to fine-tune LLMs for multi-task proficiency.

			Subset Size			
Domain	**Model**	**Composition**	$k = 100$	$k = 200$	$k = 400$	$k = 1000$
Blocksworld	GPT	pure data	**61.8±1.9%**	72.4±1.8%	81.9±1.5%	91.9±1.2%
		mixed data	58.8±1.6%	**75.4±0.6%**	**85.0±0.5%**	–
	Llama	pure data	**43.9±2.9%**	54.2±6.8%	67.6±3.3%	86.3±0.8%
		mixed data	35.1±11.6%	**58.7±4.8%**	**79.6±3.3%**	**89.6±1.5%**
Logistics	GPT	pure data	7.5±2.5%	20.6±2.1%	**45.8±3.8%**	62.3±2.3%
		mixed data	**10.4±0.1%**	**22.1±3.5%**	36.4±2.8%	–
	Llama	pure data	0.4±0.1%	10.3±0.7%	45.4±0.5%	**72.7±3.1%**
		mixed data	**5.9±1.4%**	**19.1±1.6%**	**54.0±6.4%**	68.0±5.1%

Table 5: Task solved rate in the Blocksworld and Logistics domain. k is the number of fine-tuning samples selected from one domain. E.g., $k = 100$ indicates 100 samples from Blocksworld and 100 samples from Logistics. Results are presented as mean ± standard deviation. GPT and Llama represent `GPT-3.5-turbo` and `Llama-3-8b` and their results are collected across three and five random seeds respectively. Results for GPT-3.5-turbo with $k = 1000$ are omitted in this experiment due to the high economic cost.

Combining these findings with those discussed in Section A.2 on the Data Scaling Effect, we conclude that LLMs can achieve robust System 2 planning capabilities by being fine-tuned on sufficiently large datasets composed of diverse samples from various planning domains. Our research offers valuable insights to both the research community and industry on how to effectively and efficiently train LLMs to develop strong planning capabilities.

B.5 Plan Optimality

We evaluated the optimality of the plans generated by LLMs and found that they have a high likelihood of being optimal. As shown in Table 6, GPT-3.5-turbo and Llama-3-8b achieved near-optimal performance when fine-tuned with 4,000 samples in both the Blocksworld and Logistics domains. For comparison, a human baseline in the Blocksworld domain, as reported by (Valmeekam et al., 2024b), showed that 50 human participants had a task success rate of 78.0% and a plan optimality rate of 89.7%. Given the higher complexity of the Logistics domain, the human baseline there would likely be lower than in Blocksworld. Remarkably, LLMs maintained high optimality rates—over 90% in Blocksworld and over 80% in Logistics—even when fine-tuned with just 100 samples. This result is notable and particularly relevant in the planning domain, where the optimal plan, often being the shortest, is likely to be the easiest for LLMs to identify. Achieving such superior optimality rates is challenging through prompting techniques or system designs alone, further emphasizing the importance of extensive fine-tuning. These results underscore the strong potential of LLMs in achieving System 2 planning capabilities.

B.6 Detailed Performance

Figure 10 and Figure 11 provide a detailed analysis of the LLMs' planning capabilities in the Blocksworld and Logistics domains.

Domain	Base Model	Data Composition	Subset Size $k = 100$	$k = 400$	$k = 4000$
Blocksworld	GPT-3.5-turbo	Solved Rate	61.8%	81.9%	96.5%
		Optimality Rate	89.3%	92.4%	98.4%
	Llama-3-8b	Solved Rate	43.9%	67.6%	99.3%
		Optimality Rate	90.0%	92.6%	100%
Logistics	GPT-3.5-turbo	Solved Rate	7.5%	45.8%	–
		Optimality Rate	78.3%	98.1%	–
	Llama-3-8b	Solved Rate	43.9%	67.6%	94.7%
		Optimality Rate	81.2%	91.2%	99.3%

Table 6: Analysis of the optimality rate of generated plans in Blocksworld and Logistics domain. The samples are all randomly sampled in this experiment. The Optimality rate is calculated based on the correct plans, optimality rate = (number of optimal plans)/(number of correct plans). GPT and Llama results are averaged across three and five random seeds. Results for GPT-3.5-turbo with $k = 4000$ in Logistics are omitted in this experiment due to the high economic cost.

From Figure 10, we observe that GPT's performance, as measured by the solved rate, remains consistent even as the optimal plan length increases, demonstrating robust and reliable planning capabilities. In contrast, Llama shows diminished performance on more complex tasks.

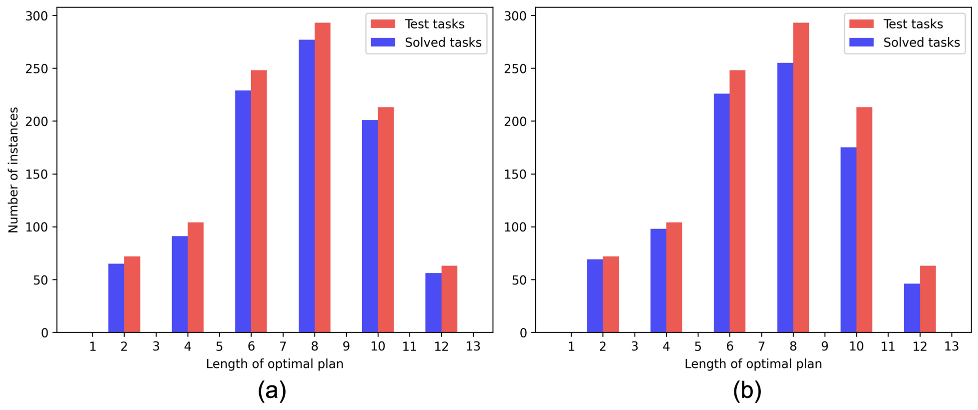


Figure 10: Distribution of the solved tasks in the Blocksworld domain. (a) GPT-3.5-turbo and (b) Llama-3-8b fine-tuned on 1000 4-block tasks and tested on 4-block tasks.

Figure 11 further reveals that LLMs can have good planning capabilities across tasks of varying complexity (i.e., they can effectively solve both short and long planning tasks). GPT's performance is worse than Llama's because it is fine-tuned on less amount of data. We do not observe a clear pattern indicating that LLMs perform better on simpler tasks (i.e., those requiring fewer steps) than on more complex tasks (i.e., those requiring more steps). These findings suggest that once LLMs are fine-tuned on sufficiently large datasets, they can perform consistently well across tasks of varying complexity in terms of plan lengths.

B.7 Language Embeddings

To generate language embeddings for planning tasks, we simplify the prompts by removing redundant information and only converting the initial and goal configurations into the embedding space to calculate the distance between tasks. Domain instructions are omitted since they are identical for all tasks. In our experiments, we used the RoBERTa (Liu, 2019) to transform natural language descriptions into the language embedding space.

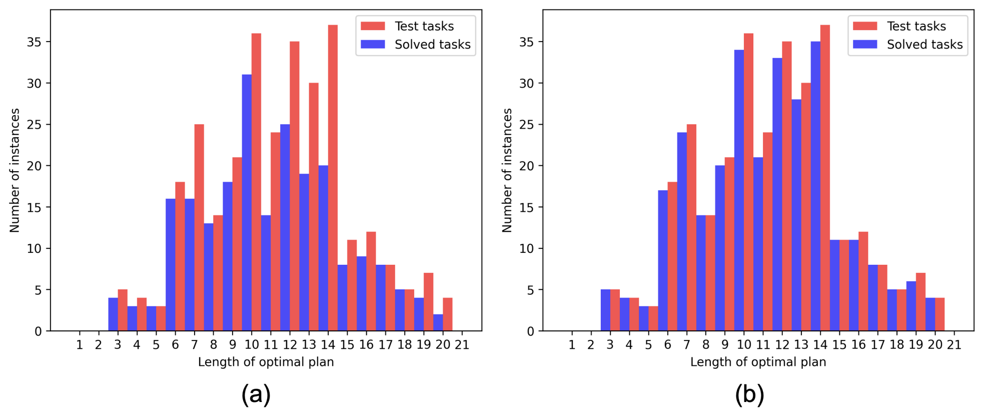


Figure 11: Distribution of the solved tasks in the Logistics domain. (a) GPT-3.5-turbo is fine-tuned on 1000 tasks, and (b) Llama-3-8b is fine-tuned on 4000 tasks.

Below is an example task in the Blocksworld domain and its altered version. The altered task slightly differs from the original task, with only the positions of the blue and yellow blocks switched. The L2 norm distance between the embeddings of these two tasks is 3.6e-07, which is too small to be effective in the downstream clustering process. This results in points within a cluster being closely packed, making it difficult to identify multiple representative points from a single cluster. This example highlights the phenomenon illustrated in Figure 4, where planning tasks converted into the language embedding space are densely packed, with clusters far apart from each other. This occurs because language embeddings are influenced more by narrative differences than by task structure.

Despite this shortcoming, CMDS-l still outperforms the Random selection, indicating the effectiveness of the Clustering-Based Maximum Diversity Sampling. Our results further validate that diversity plays an important role in automated planning, as it does in many other tasks involving the fine-tuning of LLMs.

An Example Task in the Blocksworld

Original Task

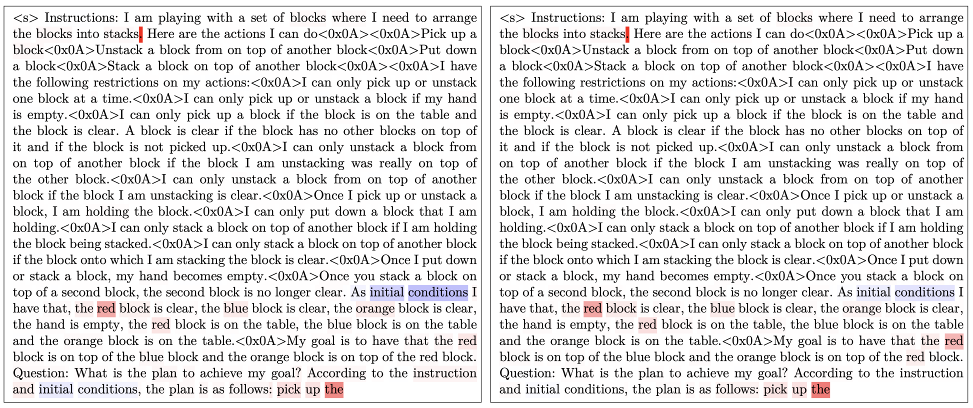
As initial conditions I have that, the red block is clear, the orange block is clear, the yellow block is clear, the hand is empty, the yellow block is on top of the blue block, the red block is on the table, the blue block is on the table and the orange block is on the table. My goal is to have that the red block is on top of the orange block and the orange block is on top of the yellow block.

Altered Task

As initial conditions I have that, the red block is clear, the orange block is clear, the blue block is clear, the hand is empty, the blue block is on top of the yellow block, the red block is on the table, the yellow block is on the table and the orange block is on the table. My goal is to have that the red block is on top of the orange block and the orange block is on top of the yellow block.

B.8 Visualized Evaluations via Attention

Beyond using task solved rates as a primary metric for evaluating model performance, we also analyze the attention mechanisms of transformers, offering a visual comparison between pre-finetuning and post-finetuning models. This analysis utilizes the Layer-wise Relevance Propagation (LRP, (Achtibat et al., 2024)) method, which integrates attention layers to provide a more detailed interpretation of the performance differences between models. An example of this visualization is shown in Figure 12. It is evident that after fine-tuning, LLMs focus more on the correct tokens, thereby increasing the likelihood of making correct decisions from the very first step.


(a) pre-finetuning

<s> Instructions: I am playing with a set of blocks where I need to arrange the blocks into stacks. Here are the actions I can do<0x0A><0x0A>Pick up a block<0x0A>Unstack a block from on top of another block<0x0A>Put down a block<0x0A>Stack a block on top of another block<0x0A><0x0A>I have the following restrictions on my actions:<0x0A>I can only pick up or unstack one block at a time.<0x0A>I can only pick up or unstack a block if my hand is empty.<0x0A>I can only pick up a block if the block is on the table and the block is clear. A block is clear if the block has no other blocks on top of it and if the block is not picked up.<0x0A>I can only unstack a block from on top of another block if the block I am unstacking was really on top of the other block.<0x0A>I can only unstack a block from on top of another block if the block I am unstacking is clear.<0x0A>Once I pick up or unstack a block, I am holding the block.<0x0A>I can only put down a block that I am holding.<0x0A>I can only stack a block on top of another block if I am holding the block being stacked.<0x0A>I can only stack a block on top of another block if the block onto which I am stacking the block is clear.<0x0A>Once I put down or stack a block, my hand becomes empty.<0x0A>Once you stack a block on top of a second block, the second block is no longer clear. As initial conditions I have that, the red block is clear, the blue block is clear, the orange block is clear, the hand is empty, the red block is on the table, the blue block is on the table and the orange block is on the table.<0x0A>My goal is to have that the red block is on top of the blue block and the orange block is on top of the red block. Question: What is the plan to achieve my goal? According to the instruction and initial conditions, the plan is as follows: pick up the

(b) post-finetuning

<s> Instructions: I am playing with a set of blocks where I need to arrange the blocks into stacks. Here are the actions I can do<0x0A><0x0A>Pick up a block<0x0A>Unstack a block from on top of another block<0x0A>Put down a block<0x0A>Stack a block on top of another block<0x0A><0x0A>I have the following restrictions on my actions:<0x0A>I can only pick up or unstack one block at a time.<0x0A>I can only pick up or unstack a block if my hand is empty.<0x0A>I can only pick up a block if the block is on the table and the block is clear. A block is clear if the block has no other blocks on top of it and if the block is not picked up.<0x0A>I can only unstack a block from on top of another block if the block I am unstacking was really on top of the other block.<0x0A>I can only unstack a block from on top of another block if the block I am unstacking is clear.<0x0A>Once I pick up or unstack a block, I am holding the block.<0x0A>I can only put down a block that I am holding.<0x0A>I can only stack a block on top of another block if I am holding the block being stacked.<0x0A>I can only stack a block on top of another block if the block onto which I am stacking the block is clear.<0x0A>Once I put down or stack a block, my hand becomes empty.<0x0A>Once you stack a block on top of a second block, the second block is no longer clear. As initial conditions I have that, the red block is clear, the blue block is clear, the orange block is clear, the hand is empty, the red block is on the table, the blue block is on the table and the orange block is on the table.<0x0A>My goal is to have that the red block is on top of the blue block and the orange block is on top of the red block. Question: What is the plan to achieve my goal? According to the instruction and initial conditions, the plan is as follows: pick up the

Figure 12: Attention visualization. The correct next token in the example is "red". The normalized attention on token 'red' for the pre-finetuning model is $\text{sum}['red'] = 0.55$ and for the post-finetuning model is $\text{sum}['red'] = 0.85$.

We then investigate how the model's attention is affected by fine-tuning on datasets selected through different methods, namely Random, CMDS-l, and CMDS-g. We randomly selected 10 examples from the hold-out test tasks, with the results summarized in Table 7. Among these examples, CMDS-g enabled the model to achieve the highest attention on the correct next token in 8 out of 10 test cases, while CMDS-l led in 2 out of 10 cases. These outcomes are consistent with the model's solved rate performance reported in previous sections. This attention visualization serves as additional evidence supporting the effectiveness of the fine-tuning process.

		Post-Finetuning		
Correct Next Token	**Pre-Finetuning**	**Random**	**CMDS-l**	**CMDS-g**
$\text{sum}['red']$	0.554	0.845	0.873	**0.928**
$\text{sum}['blue']$	0.880	0.937	**0.976**	0.967
$\text{sum}['red']$	0.602	0.881	0.936	**1.02**
$\text{sum}['blue']$	0.936	0.944	0.953	**0.998**
$\text{sum}['red']$	0.596	0.944	0.987	**1.053**
$\text{sum}['yellow']$	0.910	0.954	0.957	**0.980**
$\text{sum}['yellow']$	0.234	0.786	0.862	**0.872**
$\text{sum}['orange']$	0.317	1.117	**1.166**	1.128
$\text{sum}['orange']$	0.319	1.180	1.172	**1.187**
$\text{sum}['blue']$	0.217	0.913	0.945	**0.966**

Table 7: Attention on the correct next word for each sample index before and after fine-tuning. Base model is Llama-2-7b in this experiment.

C Domain Details and Prompts

Our implementations are based on the codebase from (Valmeekam et al., 2024b). In this section, we provide the domain properties, instructions, actions, predicates, and prompts used in our experiments.

C.1 Blocksworld Domain

C.1.1 Domain Properties

The Blocksworld domain focuses on stacking blocks on a table. One hand is available to move blocks, and only one block may be moved by the hand at a time. Blocks cannot be moved if there are blocks on top of them, and blocks cannot be stacked on a block that already has another block on top of it. The goals specify the order in which blocks within a stack should be stacked but may include multiple stacks or request that blocks be left on the table. We adopt the 4-operator version of the classic Blocksworld.

Below is the domain instruction included in the prompts to the LLMs. For one-shot prompting, the prompt contains three components: the domain instruction, an example task and its solution, the query task. For zero-shot prompting, the prompt contains two components: the domain instruction and the query task.

Domain Instruction

I am playing with a set of blocks where I need to arrange the blocks into stacks. Here are the actions I can do:

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear.
A block is clear if the block has no other blocks on top of it and if the block is not picked up.
I can only unstack a block from on top of another block if the block I am unstacking was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is clear.
Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being stacked.
I can only stack a block on top of another block if the block onto which I am stacking the block is clear.
Once I put down or stack a block, my hand becomes empty.
Once you stack a block on top of a second block, the second block is no longer clear.

Actions

1. **pick-up**: pick up the {}.
2. **put-down**: put down the {}.
3. **stack**: stack the {} on top of the {}.
4. **unstack**: unstack the {} from on top of the {}.

Predicates

1. **ontable**: the {} is on the table.
2. **clear**: the {} is clear.
3. **handempty**: the hand is empty.
4. **holding**: the hand is currently holding {}.
5. **on**: the {} is on top of the {}.

An Example Task and Its Solution in the Blocksworld Domain

```
[STATEMENT]
```
As initial conditions I have that, the blue block is clear, the hand is empty, the blue block is on top of the yellow block, the orange block is on top of the red block, the yellow block is on top of the orange block and the red block is on the table.
My goal is to have that the red block is on top of the blue block, the blue block is on top of the orange block and the yellow block is on top of the red block.

My plan is as follows:
```
[PLAN]
```
unstack the blue block from on top of the yellow block
put down the blue block
unstack the yellow block from on top of the orange block
put down the yellow block
unstack the orange block from on top of the red block
put down the orange block
pick up the blue block
stack the blue block on top of the orange block
pick up the red block
stack the red block on top of the blue block
pick up the yellow block
stack the yellow block on top of the red block
```
[PLAN END]
```

An Example Query in the Blocksworld Domain

```
[STATEMENT]
```
As initial conditions I have that, the yellow block is clear, the hand is empty, the red block is on top of the blue block, the blue block is on top of the orange block, the yellow block is on top of the red block and the orange block is on the table.
My goal is to have that the blue block is on top of the orange block, the orange block is on top of the red block and the yellow block is on top of the blue block.

My plan is as follows:
```
[PLAN END]
```

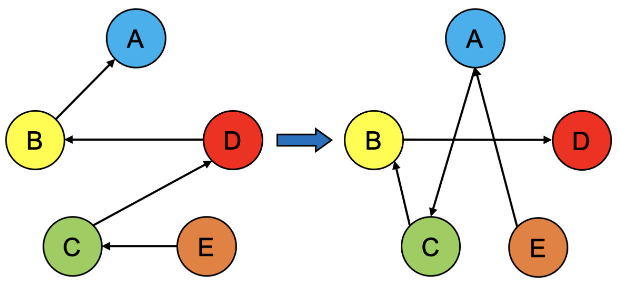
C.1.2 Graph Encoding

After representing the planning tasks using their graph structures, we proceed to encode these graphs into vectors. Below, we describe the graph encoding function employed for the Blocksworld domain in this paper. It's important to note that there are various methods to perform graph encoding, and our approach is just one of many possibilities. Drawing inspiration from (Rivlin et al., 2020; Silver et al., 2021), we designed our own graph encoding function.

As illustrated in Figure 13, our encoding converts the graph representation into a 10-dimensional vector, where a value of 1 represents a directed edge between two nodes and a value of 0 indicates no edge between the nodes. In this manner, both the initial configuration and the goal configuration are encoded as 10-dimensional vectors. To encapsulate the complete task information from both the initial and goal configurations, we concatenate these two vectors, resulting in the full vector representation of the planning task. This process converts a planning task from a natural language description to a graph representation, and finally, to a vector encoding.

Figure 13 provides the encoding method for 5-block tasks. For Blocksworld tasks with fewer blocks, we pad the non-existing nodes and edges with 0 and then add 1 to all elements in the vector. Thus, non-existing nodes and edges will have a value of 0, nodes that exist but have no edge between them will have a value of 1, and nodes with an existing edge will have a value of 2.

We have omitted the specific graph encoding method for the Logistics domain here, as it functions similarly to the approach shown in Figure 13. Readers are encouraged to design their own encoding functions tailored to their specific needs.



(a)

	Initial Configuration		Goal Configuration	
Edge	on→	on←	on→	on←
A, B	0	1	0	0
A, C	0	0	1	0
A, D	0	0	0	0
A, E	0	0	0	1
B, C	0	0	0	1
B, D	0	1	1	0
B, E	0	0	0	0
C, D	1	0	0	0
C, E	0	1	0	0
D, E	0	0	0	0

(b)

Figure 13: Encoding graphs into vectors. (a) An example task in Blocksworld. (b) Vector encoding of the example task.

C.1.3 Dataset Generation

In this section, we outline the process used to generate the dataset of planning tasks, as well as the procedures for splitting and preparing the training and testing data.

For the motivation example, which requires a large number of instances, we employed a randomization method with uniqueness checking to generate 50,000 distinct instances. For other experiments, we employ randomization to generate 128, 5000, 5000, and 5000 instances for 3-block, 4-block, 5-block, and 6-block tasks, respectively. The initial and goal configurations are randomly generated, and we ensure that each task in a dataset is distinct. For research question 1, we randomly select 100, 500, 500, and 500 tasks from each group as the test data for each task setting. Note that the test data are separated from the training data. For test data in research question 2, we uniformly sample from the four types of task settings to collectively create a testing dataset with 1000 instances, as detailed performance on different types of tasks was unnecessary for this analysis. For train data in research question 2, we sample from the remaining data to create the fine-tuning datasets, either according to CMDS algorithms or Random.

C.2 Logistics Domain

In the Logistics domain, the goal is to transport the packages to designated locations with trucks and airplanes, where trucks can only move between locations within the same city and airplanes can only fly between cities. Locations are grouped by cities. Note that there are no restrictions on the positions of the trucks and airplanes in the goal configuration. Similarly, we provide an example task and its solution from the Logistics domain below.

Domain Instruction

I have to plan logistics to transport packages within cities via trucks and between cities via airplanes. Locations within a city are directly connected (trucks can move between any two such locations), and so are the cities. In each city, there is exactly one truck and each city has one location that serves as an airport.

Here are the actions that can be performed:
Load a package into a truck.
Load a package into an airplane.
Unload a package from a truck.
Unload a package from an airplane.
Drive a truck from one location to another location.
Fly an airplane from one city to another city.

The following are the restrictions on the actions:
A package can be loaded into a truck only if the package and the truck are in the same location.
Once a package is loaded into a truck, the package is not at the location and is in the truck.
A package can be loaded into an airplane only if the package and the airplane are in the same location.
Once a package is loaded into an airplane, the package is not at the location and is in the airplane.
A package can be unloaded from a truck only if the package is in the truck.
Once a package is unloaded from a truck, the package is not in the truck and is at the location of the truck.
A package can be unloaded from an airplane only if the package in the airplane.
Once a package is unloaded from an airplane, the package is not in the airplane and is at the location of the airplane.
A truck can be driven from one location to another if the truck is at the from-location and both from-location and to-location are locations in the same city.
Once a truck is driven from one location to another, it is not at the from-location and is at the to-location.
An airplane can be flown from one city to another if the from-location and the to-location are airports and the airplane is at the from-location.
Once an airplane is flown from one city to another the airplane is not at the from-location and is at the to-location.

Actions

1. **load-truck**: load {} into {} at {}.
2. **load-airplane**: load {} into {} at {}.
3. **unload-truck**: unload {} from {} at {}.
4. **unload-airplane**: unload {} from {} at {}.
5. **drive-truck**: drive {} from {} to {} in {}.
6. **fly-airplane**: fly {} from {} to {}.
7. **drive-truck** : unstack the {} from on top of the {}.

Predicates

1. **airport**: {} is an airport.
2. **at**: {} is at {}.
3. **in**: {} is in {}.
4. **in-city**: {} is in the city {}.

An Example Task and Its Solution in the Logistics Domain

```
[STATEMENT]
```
As initial conditions I have that, location_0_0 is an airport, location_1_0 is an airport, airplane_0 is at location_0_0, package_0 is at location_1_1, truck_0 is at location_0_1, truck_1 is at location_1_0, location_0_0 is in the city city_0, location_0_1 is in the city city_0, location_1_0 is in the city city_1 and location_1_1 is in the city city_1.

My goal is to have that package_0 is at location_0_0.

My plan is as follows:
```
[PLAN]
```
drive truck_1 from location_1_0 to location_1_1 in city_1
load package_0 into truck_1 at location_1_1
drive truck_1 from location_1_1 to location_1_0 in city_1
unload package_0 from truck_1 at location_1_0
fly airplane_0 from location_0_0 to location_1_0
load package_0 into airplane_0 at location_1_0
fly airplane_0 from location_1_0 to location_0_0
unload package_0 from airplane_0 at location_0_0
```
[PLAN END]
```

An Example Query in the Logistics Domain

```
[STATEMENT]
```
As initial conditions I have that, location_0_0 is an airport, location_1_0 is an airport, airplane_0 is at location_1_0, package_0 is at location_1_1, truck_0 is at location_0_1, truck_1 is at location_1_1, location_0_0 is in the city city_0, location_0_1 is in the city city_0, location_1_0 is in the city city_1 and location_1_1 is in the city city_1.

My goal is to have that package_0 is at location_0_0.

My plan is as follows:
```
[PLAN]
```

C.3 Dataset Generation

In Logistics, four variables—number of cities, locations within a city, airplanes, and packages—affect task complexity. However, determining which of these variables has the greatest impact on task complexity and difficulty is challenging. To address this, we generated a dataset of six thousand instances with varying values for these variables. The ranges for the variables are as follows: number of cities: [2], number of locations: [2, 3], number of airplanes: [1, 2], and number of packages: [1, 2]. From this dataset, we randomly sampled three hundred instances as the test data, while the remaining instances were used as the training dataset. The test data were kept separate from the training data to ensure a clear evaluation.