PDDLPUZZLEVQA: Benchmarking Visual Planning Puzzle solving abilities using Large VLMs and Symbolic Planners

Anonymous ACL submission

Abstract

Planning is a core aspect of human intelligence. Recent planning benchmarks have proved to be challenging to a wide range of Large Language Models. Yet, planning in the context of vision has not been extensively explored. To feel this void and establish a sufficiently challenging reasoning benchmark for Vision-Language Models, we introduce PDDLPUZZLEVQA, which is a collection of $\sim 10k$ puzzles encompassing six well-known types (such as Maze-Solving, N-Queens), which explicitly require multiple-step planning to solve. We further accompany each puzzle problem with a groundtruth symbolic representation in Plan Domain Definition Language (PDDL); which in turn can be used to generate an executable plan using a symbolic planner. Therefore, we benchmark both end-to-end plan generation ability and VLM's ability to represent a planning problem presented as image and text into PDDL. Our experiments show huge deficits of state-of-the-art VLMs such as GPT4o, Gemini-flash and InternVL2.5 in all variations plan generation. Delving deeper, we analyze various syntactic and semantic errors of the VLMs while generating PDDL representation. Our dataset is the first vision and reasoning dataset to focus solely on planning puzzles, accompanied with groudtruth PDDL representation and hard benchmark for the most efficient VLMs. We plan to make both code and data publicly available for the research community.

1 Introduction

Foundational language models are trained on tasks such as next-word prediction and sequence completion. Their surprising reasoning abilities (and the so-called emergent behavior) have driven the creation of increasingly complex benchmarks in logic (Srivastava et al., 2023), math (Cobbe et al., 2021; Hendrycks et al., 2021), and commonsense reasoning (Geva et al., 2021; Sakaguchi et al., 2021; Talmor et al., 2018), though only a few remain challenging for newer Large Language Models (LLMs). Planning, a key AI discipline, remains difficult, with benchmarks like PlanBench (Valmeekam et al., 2023b) posing challenges for various models. Since end-to-end plan generation is hard and error-prone, researchers further explore combining LLMs with symbolic planners. Here, the task of LLMs is to convert natural language planning problems into Planning Domain Definition Language (PDDL) (McDermott et al., 1998), enabling the generation of verifiable, executable plans using off-the-shelf planners.

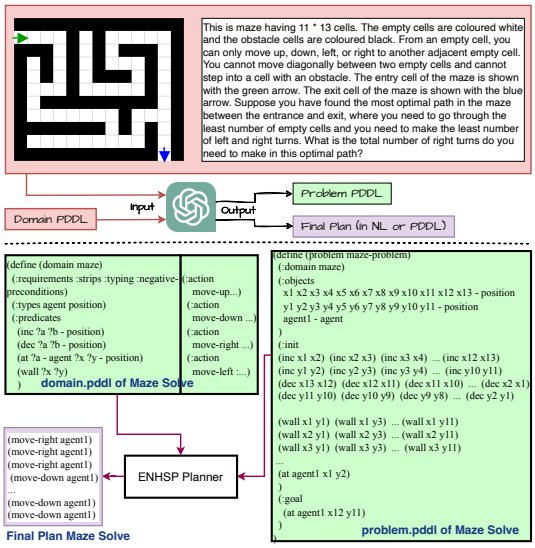


Figure 1: Overview of two task settings in PDDLPUZZLEVQA. ENHSP (Scala et al., 2016) is an external symbolic planner.

Similar to reasoning in text, multimodal reasoning benchmarks have increasingly gained attention as Vision-Language Models proliferated. Earlier benchmarks focused on evaluating visual perception and external knowledge through question answering tasks (FVQA, KB-VQA, OK-VQA)

(Wang et al., 2017b,a; Marino et al., 2019), visual commonsense reasoning (VCR, WHOOPS) (Zellers et al., 2019; Bitton-Guetta et al., 2023), spatial reasoning about objects and regions (NLVR2, CLEVR) (Suhr et al., 2019; Johnson et al., 2016b). In contrast, planning requires "thinking" and "simulating" intermediate steps given an initial state and a goal world state. Furthermore, most visual reasoning benchmarks still relies on predicting and evaluating a single final answer as an output. Planning using images has not been extensively tested.

Taking cues from the success of PlanBench and recent work on image puzzles, we specifically focus on a set of well-known single or multi-image visual puzzles that specifically requires multiple-step planning to solve. We adopt six types of puzzles from AlgoPuzzleVQA (Ghosal et al., 2024): 1) Checker-move, 2) Maze solving, 3) N-Queens, 4) Wood-slide, 5) Tower-of-Hanoi, and 6) Water-Jugs. Our goal is to benchmark the ability of Vision-Language Models to generate plans with the initial and goal states are presented as images, accompanied with limited textual description. We evaluate such ability in various stages, 1) final answer generation, 2) natural language based plan generation, and 3) generating correct symbolic problem description in PDDL language, which can be further used to generate a correct plan using any symbolic planning engine. As shown in Figure 1, we are interested in finding whether the model(s) are able to map the underlying planning problem into the desired problem PDDL. *PDDL is a standardized "Planning Domain Definition Language" (Ghallab et al., 1998; Fox and Long, 2003), widely used to describe planning domains as well as problem instances. A PDDL definition consists of two parts: domain and the problem definition*[1]. The *domain* definition presents a blueprint for representing a world in terms of *predicates* and *actions* that can be used to transition between different states of the world. The *problem* defines the *objects* present in a specific instance of the world and describes the *initial* and *goal* states.

Deviating from AlgoPuzzleVQA, we synthetically generate $9.5K$ puzzles and therefore also generate groudtruth domain and problem PDDL which is sufficient to generate the final plan, from any symbolic planner. In short, here we take a different path of solving the visual puzzles by prompting state-of-the-art VLMs to generate symbolic representation of the problem. We utilize the visual perception and symbolic code generation capabilities of VLMs and outsource the challenging task of algorithmic reasoning to an external planner (Scala et al., 2016). We make the following contributions:

1. We formulate Visual Puzzles as a Planning problem and present a benchmark process supervision dataset PDDLPuzzleVQA, comprising PDDL domain of six visual puzzles (subset of AlgoPuzzleVQA; Ghosal et al. (2024)) and $9.5K$ PDDL problems (covering 6 types of puzzles) along with templates for generating the PDDL problem files.

2. We conduct a systematic ablation with natural language, symbolic, and hybrid planning approaches and perform an in-depth analysis of different types of errors. Our results show that the largest state-of-the-art Vision-Language models (GPT4o, Gemini-flash, InternVL2.5-78B) achieve very low performance on most aspects of the task: answer generation, natural language and symbolic plan generation; demonstrating the difficulty that visual planning problems pose, and the efficacy of our benchmark.

3. Provided the limited error messages from external symbolic planner, we introduce a set of metrics to analyze different sources of syntactic and semantic errors in the generated problem PDDL programs. The semantic analysis provides detailed insights about which high-level aspects such as color, position, shape/size, or predicates make the symbolic mapping difficult.

2 Related Work

Visual Question Answering (VQA) - Over the past decade, benchmarking in vision and language has moved from testing perception and commonsense reasoning through traditional visual question answering (Antol et al., 2015; Wu et al., 2017; Goyal et al., 2017; Schwenk et al., 2022; Lu et al., 2022; Zong et al., 2024) to visual puzzle solving. In this line of reasearch, researchers test vision-language models in specific dimensions, such as visual perception, reasoning (commonsense/spatial/logical/numeric), domain or world knowledge, multi-hop reasoning (combining information and reasoning with a sequence of logical operations), planning (use of planning elements).

Aditya et al. (2016) introduced a novel visual understanding task in the form of *Image Riddles* with a total $3.3k$ samples, where each riddle has

[1] https://www.ida.liu.se/~TDDC17/info/labs/planning/writing.html

Dataset	QA-Format	Perception	World-K	Domain-K	Reasoning	Multi-Hop	Planning	PDDL
RAVEN	multi-choice	✓	✗	✗	✓	✓	✗	✗
Image Riddles	open-ended	✓	✓	✗	✓	✓	✗	✗
Visual Riddles	open-ended	✓	✓	✗	✓	✓	✗	✗
CLEVR-HYP	open-ended	✓	✗	✗	✓	✓	✗	✗
PuzzleVQA	multi-choice	✓	✗	✗	✓	✗	✗	✗
MIRB	both	✓	✓	✗	✓	✓	✗	✗
SMART-101	multi-choice	✓	✓	✗	✓	✓	✗	✗
Natural-Plan	open-ended	✗	✓	✓	✓	✓	✓	✗
PlanBench	open-ended	✗	✗	✓	✓	✓	✓	✓
Planetarium	open-ended	✗	✗	✓	✓	✓	✓	✓
AlgoPuzzleVQA	multi-choice	✓	✗	✓	✓	✓	✓	✗
PDDLPuzzleVQA (ours)	both	✓	✗	✓	✓	✓	✓	✓

Table 1: Comparison of PDDLPuzzleVQA with existing Visual Question Answering (VQA) and Planning Datasets; QA-Format - Question Answer Format (MCQ with multi-choice, questions with open-ended answers, both); Perception - Visual Perception; World-K - World Knowledge; Domain-K - Domain Knowledge; Reasoning - Commonsense/Spatial/Logical Reasoning; Planning - Need/Use of Planning Elements; PDDL - Domain and Problem specification in PDDL.

4 images and the task is to find a common concept (or word) that connects them all. Solving these riddles requires object and activity recognition (that are related to visual perception), world knowledge, commonsense, and multi-hop reasoning. Bitton-Guetta et al. (2024) present *Visual Riddles* (similar to *Image Riddles*), comprising 400 visual riddles, each featuring a unique image created by various text-to-image models, along with a question, ground-truth answer, textual hint, and attribution. CLEVR-HYP (Sampat et al., 2021) — an extension of the CLEVR dataset (Johnson et al., 2016a) — requires models to reason about hypothetical scenarios and potential outcomes based on current visual inputs. Similarly, Cherian et al. (2022) developed the SMART-101 dataset, comprising 101 unique puzzles (each puzzle has an image paired with a question), challenging models beyond straightforward visual recognition, pushing them toward more complex cognitive reasoning tasks, such as abstraction, deduction, and generalization. Similarly, RAVEN (Zhang et al., 2019) introduces the famed Raven's progressive matrices puzzle (Carpenter et al., 1990) as a dataset to test visual perception, counting, and abstraction abilities.

To understand the reasoning capabilities of large multimodal models (LMMs), Chia et al. (2024) proposed the PuzzleVQA dataset, consisting of $2k$ abstract visual puzzles that involve recognition of patterns and abstract concepts, such as colors, numbers, sizes, and shapes. Taking a step further, Ghosal et al. (2024) proposed a novel method of generating the AlgoPuzzleVQA dataset (to assess the capabilities of VLMs in solving algorithmic puzzles that require a combination of visual understanding, language comprehension, and complex reasoning). Their synthetically generated puzzles ensure correctness and scalability. Zhao et al. (2024) introduce the Multi-Image Relational Benchmark (MIRB), addressing the gap in existing evaluations, which predominantly focus on single-image inputs.

Planning - None of the VQA datasets except AlgoPuzzleVQA focus on planning, a critical aspect of human cognitive intelligence, applicable for goal-based (partially or fully specified) problem-solving. Planning involves generating a sequence of actions to transition from an initial state to a desired goal state. Traditionally, researchers in the planning domain used formal logic to represent and reason about actions, states and goals (Pelavin and Allen, 1986). For example, Planning Domain Definition Language (PDDL) utilizes formal logic to define planning problems (Ghallab et al., 1998; Fox and Long, 2003).

Inspired by traditional planning framework(s), Valmeekam et al. (2023a) introduced PlanBench, a comprehensive benchmark designed to assess the planning and reasoning capabilities of large language models (LLMs) through systematic evaluation across a diverse set of tasks, challenging LLMs in reasoning about actions and change. Through an extensive assessment of GPT3 and GPT4 (Valmeekam et al., 2023b), authors find these LLMs may not generate optimal verifiable plans, but can generate good heuristic seed plans that can be refined by integrating with external model-based planner in a LLM-modulo framework (Kambhampati et al., 2024). LLM-Modulo framework re-

Split	CM	MS	NQ	WS	TH	WJ
Test (Ghosal et al., 2024)	100	100	100	100	100	100
Train (ours)	1900	1900	1900	710	1549	1019
Test (ours)	100	100	100	100	100	100
Total (ours)	**2000**	**2000**	**2000**	**810**	**1649**	**1119**

Table 2: Dataset Statistics; Checker-Move (CM), Maze-Solving (MS), N-Queens (NQ), Wood-Slide (WS), Tower-of-Hanoi (TH), Water-Jugs (WJ)

quires both domain and problem specifications in PDDL format for tasks that involve PDDL-based planning and verification. However, LLMs may not always produce fully functional PDDL specifications and require feedback over several iterations to rectify. Another approach is to perform supervised fine-tuning of LLMs over a large parallel corpus, having problem descriptions (in natural language) and matching PDDL specifications. Zuo et al. (2024) introduced around $145,918$ text-to-problem PDDL pairs (for gripper and blocksworld domains) that can be used for aligning LLMs to generate better translations.

Although symbolic planning has proven effective in various applications, it faces several challenges (due to the requirement of comprehensive domain knowledge to define all possible states and actions within a system). This has led to a growing interest in the integration of natural language processing (NLP) with planning systems (Zheng et al., 2024). Natural language planning aims to take advantage of the flexibility and expressiveness of human language to create more adaptable and intuitive planning models. The Natural-Plan benchmark (Zheng et al., 2024) comprises the following 3 tasks - Trip Planning, Meeting Planning, and Calendar Scheduling, which have results from tools such as Google Flights, Google Maps, and Google Calendar, respectively.

We present PDDLPuzzleVQA, a dataset (having a subset of AlgoPuzzleVQA puzzles) that has different aspects, as mentioned in Table 1 to measure the problem-solving abilities of VLMs through systematic planning.

3 Constructing PDDLPuzzleVQA

We adopt the dataset generation process prescribed in Ghosal et al. (2024) and introduce suitable modifications to build the PDDLPuzzleVQA dataset. Table 2 shows the dataset statistics.

3.1 Chosen Puzzles

We consider a subset of puzzles (6 out of 18) from the AlgoPuzzleVQA dataset and formulate them as planning problems in PDDL format.

Checker-Move. This puzzle has a 1-dimensional grid of length n, where we place $n-1$ checkers of *color1* or *color2* (varying the number of checkers of each color), leaving one empty cell for the starting configuration. From a collection of m (=10) different color pairs, we sample (color1, color2) for each problem instance (§B, Figure 4). The end goal is to rearrange the checkers into a specific configuration following the rules defined in Ghosal et al. (2024) with slight modifications. We generalize the choice of color pairs and map *green* to *color1* and *red* to *color2*.

Maze-Solving. It is an $M \times N$ grid with walls, empty cells, an entry (start) point, and an exit (end) point (§B, Figure 5). An agent or a player needs to navigate starting from the entry point (denoted by a green arrow) and reach the exit point (denoted by a blue arrow) along an optimal path. The final task is to find: i) the number of left/right/total turns or ii) the number of cells in the optimal path. We maintain a precise record of the locations of walls, empty cells, starting position, and goal state to systematically evaluate the ability of VLMs to accurately identify and localize these elements during the plan generation process.

N-Queens. This is a popular 2-dimensional puzzle comprising an $N \times N$ chessboard (§B, Figure 6), where the objective is to place N queens on non-attacking positions (i.e., no two queens should share the same row, column or diagonal). Similar to AlgoPuzzleVQA (Ghosal et al., 2024), we place $N-2$ queens following the rules of the puzzle (varying N between 8 and 11), which forms the initial configuration of the puzzle. The goal is to correctly place the remaining two queens and compute the Manhattan distance between their locations. We vary the chessboard's color by sampling different color pairs, similar to Checker-Move.

Wood-Slide. The sliding block puzzle is defined on a 5×4 grid containing nine wooden blocks of varying dimensions: one 2×2, four 1×2, two 2×1, and two 1×1 (§B, Figure 7). The grid also includes two empty 1×1 spaces. Blocks are constrained within the grid and can only be moved by sliding them horizontally or vertically into adjacent empty spaces. The objective is to transform the given initial configuration into the specified goal

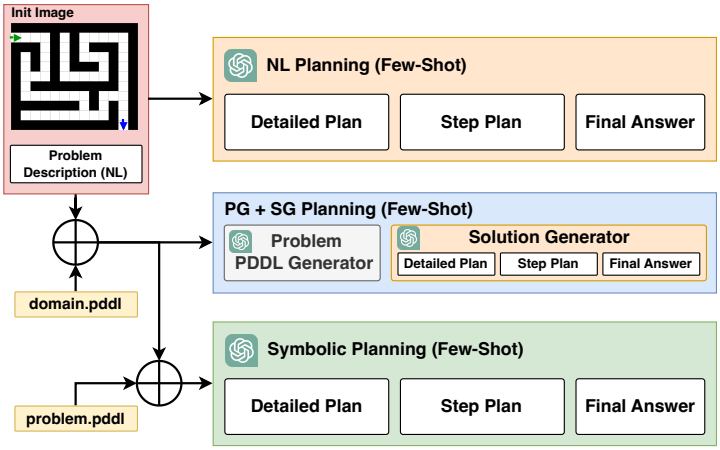


Figure 2: Illustration of different Planning Methods

configuration using the minimum number of moves, where each move consists of shifting a block by one unit in a valid direction. While our problem setup shares structural similarities with AlgoPuzzleVQA (Ghosal et al., 2024), we introduce color variations across the wooden blocks while maintaining a consistent representation of empty spaces in white to analyze the visual reasoning capabilities of VLMs. Additionally, we track the exact locations of the wooden blocks to evaluate the object detection and localization capabilities of VLMs.

Tower-of-Hanoi. The Tower of Hanoi puzzle consists of three fixed pegs and a variable number of disks ranging from three to six (§B, Figure 8). The objective is to transfer all disks from the initial configuration on the source peg to the goal configuration on the target peg, following the standard constraint that only one disk can be moved at a time and a larger disk cannot be placed on a smaller one. While structurally similar to classical Tower of Hanoi given in AlgoPuzzleVQA (Ghosal et al., 2024), we introduce color variations across the disks to analyze the visual reasoning capabilities of VLMs.

Water-Jugs. The Water Jugs puzzle requires redistributing water among jugs of varying capacities to achieve a target configuration (§B, Figure 9). Water can be transferred between jugs under two conditions: a non-empty jug can pour into a non-full jug until either the source is emptied or the destination is filled, and no water is lost during transfer. While the standard setup consists of three jugs with fixed capacities, we extend the problem to include three to seven jugs, with capacities reaching up to 15 liters. Additionally, different water colors are used to evaluate how effectively VLMs perceive and interpret the puzzle's current state.

3.2 Additional Modifications

We perform additional modifications for efficiently benchmarking perception adn planning abilities.

- **Init and Goal Images -** In the original AlgoPuzzleVQA dataset (Ghosal et al., 2024), the initial and goal states of some puzzles, such as Checker-Move, Wood-Slide, and Tower-of-Hanoi are plotted in the same image, making the visual perception task harder for VLMs. We simplify this issue by plotting separate images of init and goal states for these puzzles. We also remove textual captions, such as "Starting Configuration" and "Ending Configuration" from the images.
- **Domain PDDL -** We make use of appropriate *predicates*, *functions*, and *actions* following the specification of PDDL 2.1 (Fox and Long, 2003) and formally define the characteristics of each puzzle in a `domain.pddl` file. So, there is one `domain.pddl` file for each puzzle.
- **String Representation -** We generate a unique string representation for each puzzle to represent start/end configuration(s) and other essential parameters in a compact format.
- **Problem PDDL -** Each instance of a puzzle requires a `problem.pddl` file that encapsulates the initial and goal configurations. A `problem.pddl` file has the following important components: objects (list of objects required for representing the init and goal configurations of the puzzle), init (initial configuration of the puzzle, where necessary predicates and functions are grounded), goal (specification of the goal configuration using grounded

predicates or functions), and metric (optional section for specifying optimization criterions). It is a laborious task to manually write `problem.pddl` file for each instance of a puzzle. So, we create a puzzle-specific template that helps in generating the `problem.pddl` for each instance during dataset generation process.

• **Symbolic Plan -** For each puzzle instance, we generate a verified symbolic plan using the ENHSP planner (Scala, 2018). The planner takes the domain.pddl (puzzle-specicific) and problem.pddl (instance-specific) files as input and outputs a sequence of actions as the planning steps.

• **Natural Language Plan -** We make use of a template for each puzzle for transforming the symbolic plans into into natural language format (i.e., sequence of actions in natural language).

Please refer to Appendix §B for the detailed modifications specific to each puzzle.

4 Experiments

We evaluate different planning methods with three popular state-of-the-art vision-language models (also referred to as multimodal large-language models): GPT4o, Gemini, and InternVL2-78B. For GPT4o, we use the Azure API `gpt4o` (Model version 2024-05-13). In case of Gemini, we use the `gemini-1.5-flash` model. We also use the `InternVL2.5-78B` directly from Hugginface API[2].

For all the planning methods, we instruct VLMs (with temperature set to 0) to generate output in a specific format. The output is structured into three components: *Detailed Plan*, *Step Plan* and *Final Answer*. The *Detailed Plan* provides a structured reasoning process with step-by-step logic. The *Step Plan* provides minimalistic sequence of actions transitioning from the initial to the goal state, omitting justifications. The *Final Answer*(open-ended or multi-choice option) presents the derived solution as either an integer answer or an answer option label.

4.1 Planning Methods

As shown in Figure 2, we investigate three types of planning (few-shot setups): a) Natural Language Planning (NL Planning), b) VLM-based Problem PDDL Generation (PG) and planning with a Natural Language Solution Generator (PG + SG Planning), and c) Symbolic Planning with manually written domain and problem PDDL specifications. We adopt the eCoT prompting method from AlgoPuzzleVQA (Ghosal et al., 2024) (with an additional instruction to capture the final answer option) as a baseline.

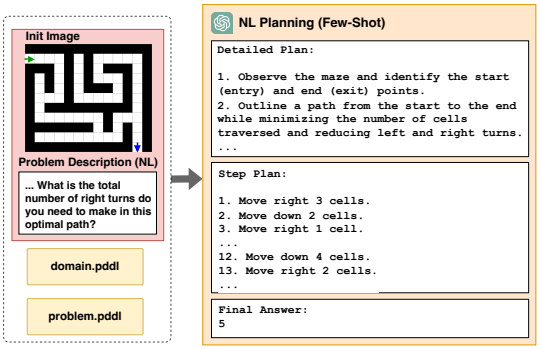


Figure 3: Example of NL Planning

• **NL Planning (Few Shot) -** Figure 3 shows the Natural Language Planning method, where we use image(s) of the initial state (and goal), followed by the problem description (in natural language) generate natural language plans by prompting VLMs. We test both 0-shot and 1-shot prompting with both in-domain (idm) and out-of-domain (oodm) samples.

• **PG + SG Planning (Few Shot) -** This planning method has two steps - i) problem PDDL Generation (PG), ii) Solution Generation (SG) (§B, Figure 13). First, we prompt a VLM-based problem PDDL Generator (with visual inputs, natural language problem description, and domain PDDL) to generate problem PDDL (in a 1-shot setting, using idm/oodm samples). The output of PG is then processed by a VLM-based Solution Generator that outputs a natural language plan, followed by a final answer.

• **Symbolic Planning (Few Shot) -** For Symbolic Planning, we employ all the available information, such as images of init/goal, natural language description, domain, and manually written problem PDDL specifications in a 1-shot setting (with idm/oodm samples) to prompt VLMs (§B, Figure 12). The generated output format aligns with the output of SG module. The primary objective of this experiment is to assess the model's ability to interpret the given domain and problem PDDL and produce solutions consistent with those generated by classical planners such as ENHSP.

[2] https://huggingface.co/OpenGVLab/InternVL2_5-78B

	0-shot		
	gpt4o	gem-fl	intern-vl2.5
Checker-Move			
eCoT (multi-choice)	17.0	-	-
NL-PL (multi-choice)	20.0	21.0	-
NL-PL (open-ended)	1.0	0.0	1.0
Maze-Solving			
eCoT (multi-choice)	30.0	-	-
NL-PL (multi-choice)	24.0	4.0	-
NL-PL (open-ended)	5.0	6.0	8.0
N-Queens			
eCoT (multi-choice)	24.0	-	-
NL-PL (multi-choice)	23.0	32.0	-
NL-PL (open-ended)	13.0	0.0	4.0
Wood-Slide			
eCoT (multi-choice)	33.0	-	-
NL-PL (multi-choice)	25.0	28.0	-
NL-PL (open-ended)	4.0	0.0	1.0
Tower-of-Hanoi			
eCoT (multi-choice)	9.0	-	-
NL-PL (multi-choice)	19.0	15.0	-
NL-PL (open-ended)	2.0	0.0	5.0
Water-Jugs			
eCoT (multi-choice)	6.0	-	-
NL-PL (multi-choice)	29.0	11.0	-
NL-PL (open-ended)	11.0	2.0	10.0

Table 3: Comparison of solving visual puzzles using different state-of-the-art Vision Language Models (VLMs) such as gpt4o, `gemini-1.5-flash` (gem-fl), intern-vl (int-vl), NL-PL: Natural Language Plan + Final Answer

4.2 Results of 0-Shot Prompting

We prompt GPT4o and Gemini-Flash to provide detailed instructions and ask them to produce a detailed plan summarizing the logic behind solving the puzzle, followed by a step-by-step final plan. In the multiple-choice question setting, we also ask to output the option. An example prompt is shown in Table 11. For open-ended setting, the model is simply asked to generate the final answer (Table 8). `Intern-VL2.5-78B` is benchmarked only in open-ended setting. In some cases, `InternVL2.5` does not even produce the final response.

Results. Our results from Table 3 show extremely low (below 5%) performance in the open-ended setting for all puzzles except N-Queens and Water-Jugs. The multiple-choice option raises the accuracy of GPT4o and Gemini-Flash in all cases. GPT4o clearly outperforms other models in most cases, except Gemini-Flash outperforming in N-Queens and and Wood-slide. The best performance of Intern-VL2.5 is 10% in Water Jugs (lagging behind GPT4o by 1%). For Maze-solving and Wood-slide, asking GPT4o to explain the logic increases performance by 6% & 8% resp.ly. This strategy leads to a decrease for three puzzle types, while keeping the performance somewhat similar for N-Queens.

	gpt4o		gem-fl	
	PG+SG	Sym	PG+SG	Sym
Checker-Move	1.0	14.0	1.0	21.0
Maze-Solving	5.0	9.0	3.0	10.0
N-Queens	11.0	30.0	6.0	3.0
Wood-Slide	2.0	29.0	0.0	5.0
Tower-of-Hanoi	0.0	16.0	0.0	6.0
Water-Jugs	9.0	28.0	1.0	13.0
AVG	4.67	21	1.83	9.67

Table 4: Accuracy of PG + SG Planning vs Symbolic (Sym) Planning (1-shot-idm setting); gem-fl: `gemini-1.5-flash` ; AVG: Average; idm: in-domain.

4.3 Results of 1-shot Prompting

Because of poor performance of `InternVL2.5`, we explore one-shot setting for GPT4o and Gemini-flash. We create two settings, where the in-context example comes from the same type of puzzle (`idm`) and another where the example comes from a different puzzle type (`oodm`). The second setting provides some indication of results on completely unseen puzzles.

Results. Table 6 shows the effect of in-context examples. Interestingly, the `idm` and the (`oodm`) setting improves `gpt4o` performance by large margins for many puzzle types. For example, Water Jugs performance with `idm` by 40% and with `oodm` by 25%. `idm` decreases performance slightly for Checker-Move and Maze-Solving (by 1%), while `oodm` improves them by 11% and 8% respectively. These performance improvements are limited for the eCOT setting with multi-choice answers. For other two settings of direct plan and solution generation, in-context examples do not show any improvement. Similarly, `gemini-flash` lags behind `gpt4o` for all puzzles except N-queens in the `oodm` setting. From Table 4, we observe that Symbolic Planning significantly outperforms PG+SG Planning, underscoring the importance of improving VLMs to generate correct problem PDDL specifications for solving planning problems.

4.4 Semantic Errors in Problem PDDL

Potential semantic errors in the generated problem PDDL specification (init / goal) originate from an incorrect visual understanding of a puzzle.

Incorrect Specification of Init/Goal

Error Type	Error Name	Checker-Move		Maze-Solving		Wood-Slide		Tower-of-Hanoi		Water-Jugs		N-Queens*	
		idm	oodm	idm	oodm	idm	oodm	idm	oodm	idm	oodm	idm	oodm
Syntax	Undefined Entities	0	3	0	1	1	9	2	56	0	0	0	1
	General Syntax Errors	0	1	0	0	2	5	0	0	0	17	2	1
	Duplicate Entities	1	11	0	13	0	0	0	0	0	0	0	0
	Inconsistent Parameter Use	0	3	0	9	0	0	0	0	0	1	0	0
	Incomplete	0	0	0	6	3	0	0	0	0	0	4	2
	Other	6	19	14	18	9	18	19	25	3	4	0	0
Semantic	**Incorrect Spec. of Init**												
	Color	76	52	-	-	-	-	-	-	-	-	-	-
	Position	71	50	86	49	84	52	46	2	0	0	10	10
	Shape/Size/Count	3	0	24	25	18	26	79	19	69	58	0	0
	Predicates/Functions	3	0	0	0	0	0	45	2	0	0	10	10
	Incorrect Spec. of Goal												
	Color	83	54	-	-	-	-	-	-	-	-	-	-
	Position	87	59	85	53	84	52	42	2	0	0	0	0
	Shape/Size/Count	1	0	0	0	71	48	0	0	59	55	0	0
	Predicates/Functions	0	0	0	0	0	0	20	2	0	0	0	6
	Metric	0	0	0	0	1	9	0	0	0	27	0	2

Table 5: Error Analysis of the PDDL Generation (PG) module; we analyze 100 samples for each puzzle, except N-Queens* (10 samples).

- **Color -** Some puzzles (such as Checker-Move) make use of colored objects. In such cases, wrong identification of colors leads to incorrect specification of the init and/or goal states.
- **Position -** This error is common for puzzles that require absolute or relative position of objects, which gets reflected via incorrect specification of predicates and/or functions in init and/or goal.
- **Shape/Size/Count -** In addition to color and position, other important aspects related to visual perception are shape, size and count of objects. We group all of them under one category - shape/-size/count and inspect such errors in the init/goal sections of the generated problem PDDL files.
- **Predicates/Functions -** This error is related to incorrect usage of predicates/functions.
- **Metric -** Some of the puzzles require the declaration of an optimization metric in the problem PDDL. We flag a metric error whenever the metric specification is absent in the generated problem PDDL file.

Please refer to §C in the Appendix for the definitions of Syntax Errors in Problem PDDL.

Observations. Table 5 provides insights about the different errors in the generated problem PDDL of the PG module. Interestingly more than 80% errors are made while identifying the correct color or position, whenever such information is required for accurate problem formulation. VLMs make less but considerable errors in identifying shape, size and count (below 50% in most cases). Such errors are possibly attributed to the "binding problem" (Campbell et al., 2024). VLMs make the least amount of errors in identifying common predicates and functions, possibly owing to limited linguistic variability amongst the puzzle types. From a syntactic point of view, `gpt4o` suffers from using many undefined entities for Tower-of-Hanoi (56% errors in `oodm` setup). Similarly there are considerable errors while mapping same objects, predicates or constants to separate symbolic entities. In summary, this analysis shows VLMs have limited perception ability in the context of puzzles, limiting their applicability.

5 Conclusion

Inspired from the usefulness of the recent planning benchmarks in evaluating large language models, we introduce PDDLPuzzleVQA, a $9.5k$ visual puzzles dataset, explicitly encompassing six planning puzzle types for benchmarking vision-language models. Each puzzle is accompanied with input images, and textual context, a groundtruth planning problem description in a popular planning logical language (PDDL), and a groundtruth symbolic and a tempated natural language plan. Our results show the largest state-of-the-art Vision-Language models (VLMs) lack in almost all aspects of the problem, such as natural language or symbolic plan generation. Such VLMs also do not show efficiency in a VLM-modulo framework, where VLM's task is to generate the problem and domain PDDL, which can be used to generate the final plan using an external planner. Delving deeper, we find VLMs face high volume of errors in identifying color, shape, position while mapping the problem symbolically (possibly reaffirming the binding problem).

Limitations

Our work is the first to introduce an exclusively visual planning puzzles dataset, where each puzzle is accompanied by a groundtruth symbolic plan domain and problem description, which enables the benchmarking of both VLM and VLM-Modulo frameworks. Current work has the following limitations:

- Our dataset focuses on six well-known types of planning problems, which somewhat limits the diversity. Future work should extend the types of planning problems, focusing on increasing the problem complexity.
- The current dataset is exclusively in English, as we want to specifically focus on visual perception and reasoning abilities. In future, we plan to introduce multilingual versions of the dataset.
- Currently, we have not experimented with any finetuning approaches to improve VLMs due to resource constraints. We believe that, to build better puzzle understanding, we may need to involve various reasoning objectives during supervised or instruction finetuning stages of VLM training. This will, however, require significant hardware resources.

References

Somak Aditya, Yezhou Yang, Chitta Baral, and Yiannis Aloimonos. 2016. Answering image riddles using vision and reasoning through probabilistic soft logic. *Preprint*, arXiv:1611.05896.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proceedings of the 2015 International Conference on Computer Vision*, ICCV '15.

Nitzan Bitton-Guetta, Yonatan Bitton, Jack Hessel, Ludwig Schmidt, Yuval Elovici, Gabriel Stanovsky, and Roy Schwartz. 2023. Breaking common sense: Whoops! a vision-and-language benchmark of synthetic and compositional images. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2616–2627.

Nitzan Bitton-Guetta, Aviv Slobodkin, Aviya Maimon, Eliya Habba, Royi Rassin, Yonatan Bitton, Idan Szpektor, Amir Globerson, and Yuval Elovici. 2024. Visual riddles: a commonsense and world knowledge challenge for large vision and language models. *Preprint*, arXiv:2407.19474.

Declan Campbell, Sunayana Rane, Tyler Giallanza, Nicolò De Sabbata, Kia Ghods, Amogh Joshi, Alexander Ku, Steven M Frankland, Thomas L Griffiths, Jonathan D Cohen, et al. 2024. Understanding the limits of vision language models through the lens of the binding problem. *arXiv preprint arXiv:2411.00238*.

Patricia A. Carpenter, Marcel A. Just, and Peter Shell. 1990. What one intelligence test measures: a theoretical account of the processing in the Raven progressive matrices test. *Psychological Review*, 97(3):404–431.

Anoop Cherian, Kuan-Chuan Peng, Suhas Lohit, Kevin A Smith, and Joshua B Tenenbaum. 2022. Are deep neural networks smarter than second graders? *Preprint*, arXiv:2212.09993.

Yew Ken Chia, Vernon Yan Han Toh, Deepanway Ghosal, and Soujanya Poria. 2024. Puzzlevqa: Diagnosing multimodal reasoning challenges of language models with abstract visual patterns. *Preprint*, arXiv:2403.13315.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems.

M. Fox and D. Long. 2003. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

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

Deepanway Ghosal, Vernon Toh Yan Han, Chia Yew Ken, and Soujanya Poria. 2024. Are language models puzzle prodigies? algorithmic puzzles unveil serious challenges in multimodal reasoning.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the 2017 Conference on Computer Vision and Pattern Recognition*, CVPR '17, pages 6904–6913.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2016a. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *Preprint*, arXiv:1612.06890.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2016b. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997.

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.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: multimodal reasoning via thought chains for science question answering. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 1–15.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.

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

R. Pelavin and J.F. Allen. 1986. A formal logic of plans in temporally rich domains. *Proceedings of the IEEE*, 74(10):1364–1382.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Shailaja Keyur Sampat, Akshay Kumar, Yezhou Yang, and Chitta Baral. 2021. Clevr_hyp: A challenge dataset and baselines for visual question answering with hypothetical actions over images. *Preprint*, arXiv:2104.05981.

Enrico Scala. 2018. The ENHSP Planning System. Accessed: 2025-02-09.

Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramirez. 2016. Interval-based relaxation for general numeric planning. In *ECAI 2016*, pages 655–663. IOS Press.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. 2022. A-okvqa: A benchmark for visual question answering using world knowledge. In *Proceedings of the 17th European Conference on Computer Vision*, pages 146–162.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, and et al. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, pages 1–95.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. Planbench: an extensible benchmark for evaluating large language models on planning and reasoning about change. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pages 1–13.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models : A critical investigation. *Preprint*, arXiv:2305.15771.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Henge. 2017a. Explicit knowledge-based reasoning for visual question answering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1290–1296. AAAI Press.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. 2017b. Fvqa: Fact-based visual question answering. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2413–2427.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. 2019. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR '19, pages 5312–5322.

Bingchen Zhao, Yongshuo Zong, Letian Zhang, and Timothy Hospedales. 2024. Benchmarking multi-image understanding in vision and language models: Perception, knowledge, reasoning, and multi-hop reasoning. *Preprint*, arXiv:2406.12742.

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.

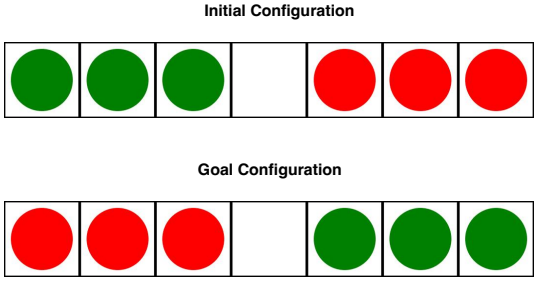
Yongshuo Zong, Tingyang Yu, Ruchika Chavhan, Bingchen Zhao, and Timothy Hospedales. 2024. Fool your (vision and) language model with embarrassingly simple permutations. *Preprint*, arXiv:2310.01651.

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

A Additional Results Table

	1-shot-idm		1-shot-oodm	
	gpt4o	gem-fl	gpt4o	gem-fl
Checker-Move				
NL-PL (multi-choice)	16.0	28.0	28.0	18.0
NL-PL (open-ended)	3.0	4.0	1.0	0.0
PG + SG-PL (open-ended)	1.0	1.0	4.0	1.0
Maze-Solving				
NL-PL (multi-choice)	29.0	36.0	38.0	34.0
NL-PL (open-ended)	8.0	11.0	11.0	11.0
PG + SG-PL (open-ended)	5.0	3.0	9.0	9.0
N-Queens				
NL-PL (multi-choice)	26.0	27.0	18.0	25.0
NL-PL (open-ended)	7.0	1.0	7.0	0.0
PG + SG-PL (open-ended)	11.0	6.0	7.0	6.0
Wood-Slide				
NL-PL (multi-choice)	34.0	33.0	32.0	17.0
NL-PL (open-ended)	9.0	13.0	5.0	0.0
PG + SG-PL (open-ended)	2.0	0.0	1.0	0.0
Tower-of-Hanoi				
NL-PL (multi-choice)	27.0	19.0	18.0	15.0
PL PL (open-ended)	5.0	0.0	3.0	0.0
PG + SG-PL (open-ended)	0.0	0.0	0.0	0.0
Water-Jugs				
NL-PL (multi-choice)	46.0	22.0	31.0	11.0
NL-PL (open-ended)	25.0	4.0	13.0	2.0
PG + SG-PL (open-ended)	9.0	1.0	13.0	1.0

Table 6: Comparison of solving visual puzzles using different state-of-the-art Vision Language Models (VLMs) such as gpt4o, gemini-1.5-flash (gem-fl), intern-vl (int-vl) in a 1-shot setting with in-domain (idm) and out-of-domain (oodm) samples as few-shot examples; NL-PL: Natural Language Plan + Final Answer; PG + SG-PL: Problem PDDL Generation followed by Solution Generator-based Planning



A checker game is being played on a grid of 7 squares with 3 green and 3 red checkers. Initially, the checkers are arranged as shown in the starting configuration with the 6 checkers occupying 6 squares and one unoccupied square. Green checkers only move rightward and Red checkers only move leftward. Every move is either i) a slide to the adjacent empty square, or ii) a jump over one position to an empty square, provided the checker being jumped over is of a different colour. Each square can accommodate a maximum of one checker at any time. How many moves are required to reach the ending configuration from the starting configuration following the specified rules?

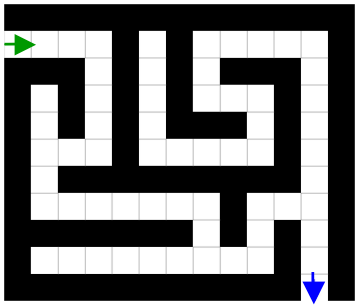
Gold Answer: 15

Figure 4: Checker-Move Puzzle

B Detailed Modifications

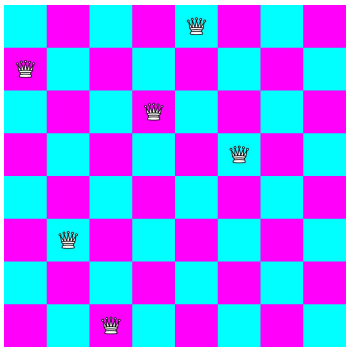
We use a tailored set of constraints to vary different visual elements of each puzzle during dataset generation and store a string representation that can be used for future research.

- **Checker-Move -** For this puzzle, the constraint set includes grid length (L = 5 to 20), color pairs (10 variations), configurations of checker counts for each color, and varying empty square positions. We represent color1 and color2 by digits 1 and 2, and empty cells by 0. Each sample contains an initial and goal state string, separated by a new line. An example of init state string: "1110222".
- **Maze-Solving -** We generate configurations by systematically varying start and end cell placements within four defined regions of the maze, along with start-end color pairs, object shapes, and wall colors. Here empty cells (0), walls (1), start position (S), and end position (E) are represented in a row-wise format separated by '\n' characters. We record moves in a path using the characters 'l' (left) and 'r' (right). An example of init state string:
"1111111111111
S000101000001
1110101011101
1010101000101
1010101110101
1000100000101
1011111111101
1000000010001
1111111010101
1000000000101
11111111111E1"
- **N-Queens -** We introduce constraints by varying chessboard size (N=8 to 11), color pairings, and queen placements. We denote each queen using the character 'Q', and use digits 1 and 2 for color1 and color2, respectively. An example of init state string:
"1212Q212
Q1212121
121Q1212
21212Q21
12121212
2Q212121
12121212
21Q12121"



This is maze having 11 * 13 cells. The empty cells are coloured white and the obstacle cells are coloured black. From an empty cell, you can only move up, down, left, or right to another adjacent empty cell. You cannot move diagonally between two empty cells and cannot step into a cell with an obstacle. The entry cell of the maze is shown with the green arrow. The exit cell of the maze is shown with the blue arrow. Suppose you have found the most optimal path in the maze between the entrance and exit, where you need to go through the least number of empty cells and you need to make the least number of left and right turns. What is the total number of right turns do you need to make in this optimal path? Gold Answer: 5

Figure 5: Maze-Solving Puzzle



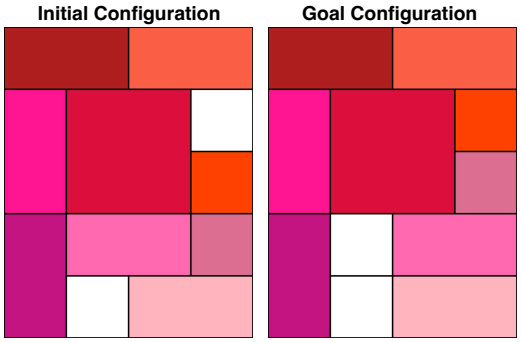
You are given an 8 * 8 chessboard. The Manhattan distance between two squares in a chessboard is equal to the minimal number of orthogonal King moves between these squares on the otherwise empty board. The objective is to place 8 chess queens on this board so that no two queens threaten each other; i.e. no two queens share the same row, column, or diagonal. 6 queens have already been placed in some of the squares of the board, as shown in the image. Suppose you pick two squares to place the two remaining queen pieces in a way that fulfills the objective. What is the Manhattan distance between these two squares? Gold Answer: 3

Figure 6: N-Queens Puzzle

• **Wood-Slide -** We vary the grid dimensions, block count, and the position and color of empty blocks to increase the diversity of the samples. Here we identify each block using block ID numbers, color codes are mapped to each unique block ID, and each row is separated by '\n' with 2 separate strings for the initial and goal state. An example of init and goal state strings:
Init
"2,2,3,3
6,1,1,0
6,1,1,4
7,8,8,5
7,0,9,9"
Goal
"2,2,3,3
6,1,1,4
6,1,1,5
7,0,8,8
7,0,9,9"

• **Tower-of-Hanoi -** For this puzzle, we vary the number of disks, disk and rod colors, and rod heights. We denote the pegs or rods using characters: 'A', 'B', and 'C' sequentially, separated by '\n' with disks size represented by numbers (where smaller numbers denote smaller disks). Disk positions on each rod are listed in order from bottom to top, separated by '>'. An example of init and goal state strings:
Init
"A=6>5>2>1
B=3
C=4"
Goal
"A=6>5
B=0
C=4>3>2>1"

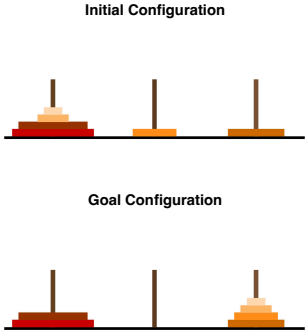
• **Water-Jugs -** Here we vary the number of jugs, capacity, volume of liquid (in each jug), and color of both jugs and liquid. Water Jugs employs strings for both start and end states, with jugs separated by '\n' and defined by current volume/capacity ratios, with color hex codes specified for the liquid. An example of init state string:
Init
"A=12/13
B=10/12
C=5/6"
Goal
"A=13/13
B=10/12
C=4/6"



Consider a sliding block puzzle of grid size 5 * 4 units. It has 9 wooden blocks of varying sizes: one 2 * 2, four 1 * 2, two 2 * 1, and two 1 * 1. The gird also has two empty 1 * 1 spaces. The blocks cannot be removed from the grid, and may only be slid horizontally and vertically within its boundary. A move is defined as selecting a block that is slideable, and moving it by 1 unit either horizontally or vertically, whichever is possible. The images show the starting and ending configurations of the puzzle grid. The wooden blocks are shown in various shades of brown and the empty spaces are shown in white. What is the minimum number of moves required to reach the ending configuration from the starting configuration?

Gold Answer: 3

Figure 7: Wood-Slide Puzzle



You are playing a Tower of Hanoi game with 3 rods and 6 disks of various diameters, which can slide onto any rod. You are given the starting and ending configuration of the game as shown in the top and the bottom of the image, respectively. The game has the following rules: i) Only one disk may be moved at a time; ii) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod; and iii) No disk can be placed on top of a disk that is smaller than it. What is the minimum number of moves required to go from the starting to the ending configuration?

Gold Answer: 4

Figure 8: Tower-of-Hanoi Puzzle

Color - We store the hex color codes used for rendering different elements of each puzzle. This ensure that the dataset captures both structural and visual diversity.

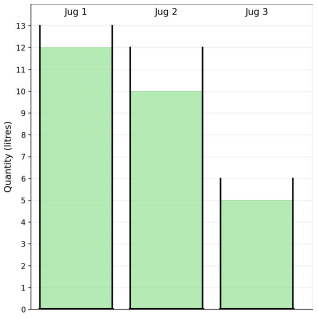
C Syntax Errors in PDDL Problems

- **Duplicate Entities -** Occurs when the same object, predicate, function, or constant is mentioned more than once in problem PDDL.
- **Undefined Entities -** Use of predicates, functions, object types that are not defined in domain PDDL.
- **Inconsistent Parameter Use -** Use of a predicate/function with either a parameter of an incompatible type or a different number of parameters than it was defined with.
- **General Syntax Errors -** These are generic syntax errors that do not align with the specification of the PDDL language.
- **Incomplete -** Occurs when problem PDDL specification is incomplete.
- **Other -** We label all other syntax errors as *Other*.

D Planning Complexity

- **Planning Time -** According to the implementation of the ENHSP-Planner[3], planning time corresponds to the total time spent exploring the state space to find a solution plus pre- and post-processing time.
- **Expanded Nodes -** This refers to the number of nodes (or states) that the planner has fully expanded during the search process. Expanding a node means generating all its successor states by applying actions. This metric provides insight into the efficiency of the search; fewer expanded nodes typically indicate a more efficient search.
- **States Evaluated -** This represents the number of states evaluated by the planner's heuristic function during the search. Some of these states may have been expanded, while others may have been pruned or discarded based on heuristic estimates. This number is often higher than the number of expanded nodes because heuristics may evaluate states without necessarily expanding them; it helps to analyze how extensively the heuristic function was used. If the number of evaluated states is much higher than the expanded nodes, the heuristic may be computationally expensive.

[3] https://gitlab.com/enricos83/ENHSP-Public



You are given 3 jugs of capacities 13, 12, 6 litres. Initially, the amount of water that is contained in each jar is shown in the image. A single step of water pouring from one jug to another is constrained by the following rules: i) take a non-empty jug and pour water from it to another non-full jug until the first one becomes empty or the second one becomes full; and ii) no water can be spilt while pouring. The objective is to reach the amounts of 13, 10, 4 litres of water in the jugs from left to right, respectively. What is the minimum number of water pouring steps required to achieve the objective?

Gold Answer: 1

Figure 9: Water-Jugs Puzzle

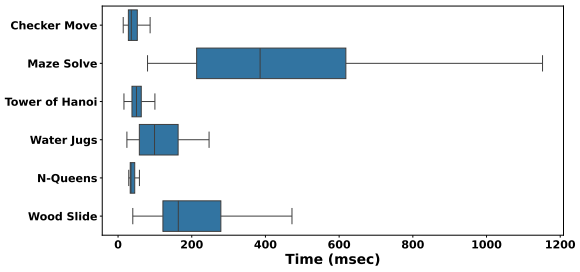


Figure 10: Variation in Planning Time

E Examples of Planning Methods

Figures 12 and 13 demonstrate the working of Symbolic PG + SG Planning methods, respectively.

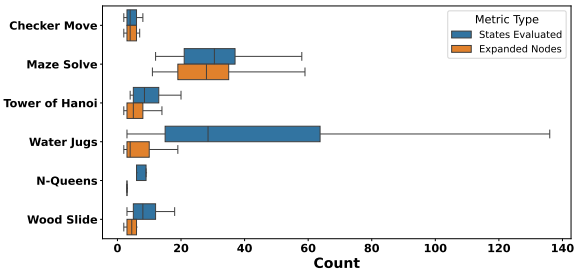


Figure 11: Variation in Problem Complexity

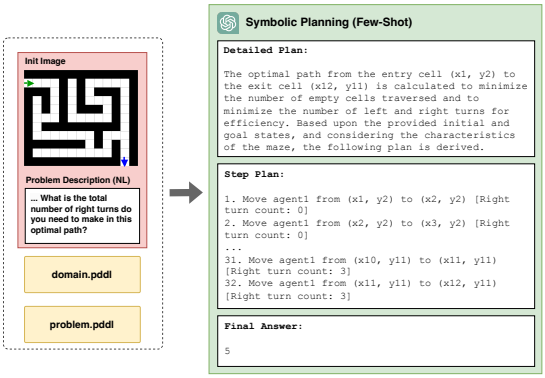


Figure 12: Example of Symbolic Planning

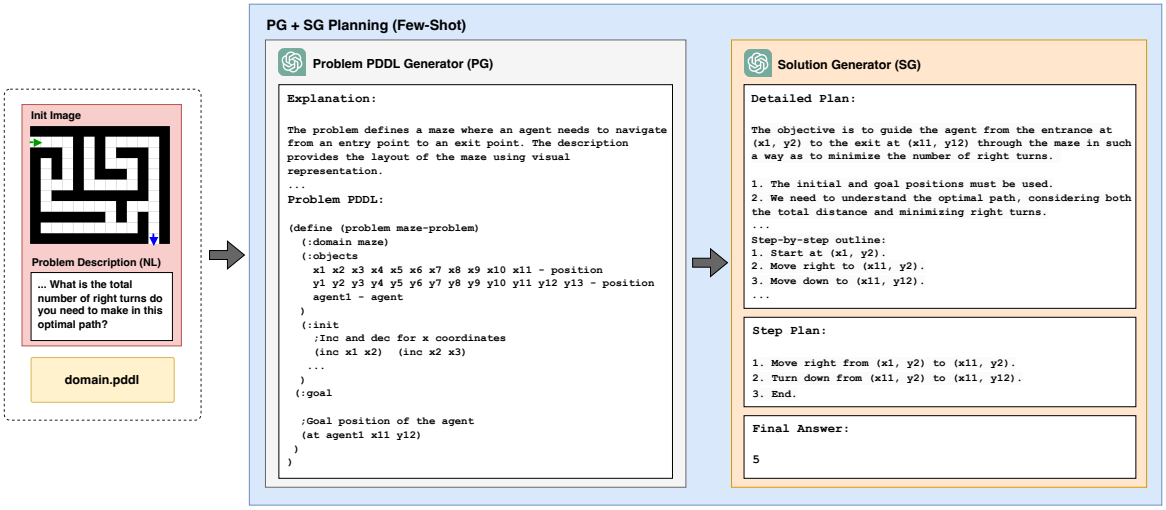


Figure 13: Example of PG + SG Planning

Instruction

```
You are a friendly and helpful assistant adept at understanding visual puzzles and formulating detailed plans in natural language.
Your task is to analyze the provided puzzle, derive the correct answer, and select the corresponding option from the
provided multiple-choice options ["A", "B", "C", "D"].
**Instructions**:
- Provide your response in the following format:
```
<ANSWER>
Detailed Plan:
[Detailed plan explaining the logic behind solving the puzzle.]
Step Plan:
[Step-by-step plan without explanations.]
Correct Option:
[The single correct option from ["A", "B", "C", "D"] that matches derived answer.]
</ANSWER>
```
- Ensure the plan and answer are clearly separated and labeled as shown.
- Under **Detailed Plan**, include the logic behind each step, explanations for why each step is taken, and any changes in
state resulting from that step. You may also describe the initial and goal states to provide context and reasoning.
- Under **Step Plan**, list only the steps that need to be taken, without any explanations or additional information.
- Under **Correct Option**, add only that **option** from ("A", "B", "C", or "D") where the derived answer matches the answer
in corresponding option given. Don't add any additional text, integer answer, or any option apart from ["A","B","C","D"].
```

In-Context Examples

Question: *A checker game is being played. . . How many moves are required to reach ending configuration from starting configuration following specified rules?*

Options:*['A': 5, 'B': 6, 'C': 1, 'D': 3]*

images : [init and goal state images. . .]

Answer:

Step Plan:

1. The yellow checker c10 jumps over a blue checker from position p11 to land at the empty position p9, with the blue checker in p10.

2. The blue checker c9 moves from position p10 to the adjacent empty position p11 to the right.

3. The blue checker c8 jumps over a yellow checker from position p8 to land at the empty position p10, with the yellow checker in p9.

Correct Option: *D*

Input Question

Question: *A checker game is being played on a grid of 12 squares with 7 blue and 4 yellow checkers. Initially, the checkers are arranged as shown in the starting configuration with the 11 checkers occupying 11 squares and one unoccupied square. Blue checkers only move rightward and Yellow checkers only move leftward. Every move is either i) a slide to the adjacent empty square, or ii) a jump over one position to an empty square, provided the checker being jumped over is of a different colour. Each square can accommodate a maximum of one checker at any time. How many moves are required to reach the ending configuration from the starting configuration following the specified rules?*

Options:*['A': 4, 'B': 5, 'C': 2, 'D': 3]*

images :

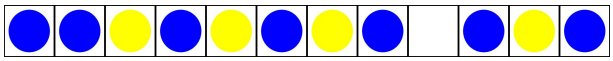


Figure: Initial Configuration

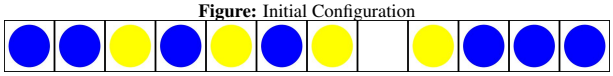


Figure: Goal Configuration

Output

Detailed Plan: *The key to solving this puzzle is to follow the movement rules for blue and yellow checkers while taking into account their allowed directions and whether they can slide to an adjacent empty square or jump over another. We need to ensure each checker ends up in its respective position, shifting the empty square around strategically.*

Observations:

- The empty square starts at position 9.

- Blue checkers in positions 1-2-3-6-8-11-12 must all be moved one step to the right.

- Yellow checkers in positions 4-5-7-9-10 must all be moved one step to the left.

- The trick is mainly to facilitate the movement of blue checkers into positions originally occupied by yellow checkers and vice versa.

Step Plan:

1. Move the blue checker from position 6 to the empty square (position 9).

2. Move the yellow checker from position 7 to position 6 (newly vacated).

3. Move the blue checker from position 8 to the empty square (position 7).

4. Move the yellow checker from position 10 to position 9.

Correct Option: *D*

Table 7: NL Planing (multi-choice) using GPT4o in 1-shot incontext setting for checker move puzzle

Instruction

```
You are a friendly and helpful assistant adept at understanding visual puzzles and formulating detailed plans in natural language.
Ensure your answers are comprehensive and provide clear instructions to solve the puzzle.

**Instructions**:
- Provide your response in the following format:
  ```
  <ANSWER>
  Detailed Plan:
  [Detailed plan]

  Step Plan:
  [Step-by-step plan]

  Final Answer:
  [Final answer]

  </ANSWER>
  ```
- Ensure the plan and answer are clearly separated and labeled as shown.
- Under **Detailed Plan**, include the logic behind each step, explanations for why each step is taken, and any changes in
state resulting from that step. You may also describe the initial and goal states to provide context and reasoning.
- Under **Step Plan**, list only the steps that need to be taken, without any explanations or additional information.
```

In-Context Examples

Question: *You are playing a Tower of Hanoi game with 3 rods and 5 disks. What is the minimum number of moves required?*

images : [init and goal state images. . .]

Answer:

Step Plan:

Move disc d1 from the top of disc d4 to the top of disc d2

Final Answer: *1*

Input Question

Question: *You are playing a Tower of Hanoi game with 3 rods and 4 disks. What is the minimum number of moves required?*

images :

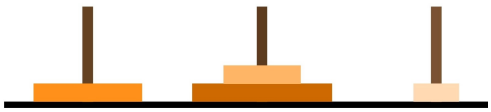


Figure: Initial Configuration

Figure: Goal Configuration

Output

Detailed Plan: *The initial state has 4 disks on 3 different rods. The goal is to arrange these disks on the third rod following the Tower of Hanoi rules:*

1. Only one disk can be moved at a time.

2. Each move involves taking the top disk from one of the stacks and placing it on top of another stack or on an empty rod.

3. No larger disk may be placed on top of a smaller disk.

4. Move disk from peg 3 to peg 2.

5. Move disk from peg 1 to peg 2.

6. Move disk from peg 1 to peg 3.

The given state:

- Rod 1: 1 large disk

- Rod 2: 2 medium disks

- Rod 3: 1 small disk

The goal state:

- Rod 1: Empty

- Rod 2: 1 medium disk

- Rod 3: 1 large disk, 1 medium disk, 1 small disk

To solve this, we need to consider repositioning the disks to achieve the goal state efficiently. We will move the smallest disk first from rod 3.

Step Plan:

1. Move D4 from Rod 3 to Rod 2

2. Move D3 from Rod 2 to Rod 3

3. Move D4 from Rod 2 to Rod 1

4. Move D2 from Rod 2 to Rod 3

5. Move D4 from Rod 1 to Rod 3

Final Answer: *5*

Table 8: NL Planning (open-ended) using GPT4o in 1-shot incontext setting for tower of hanoi puzzle

Instruction

You are a friendly and helpful assistant adept in understanding visual puzzles and formulating them as planning problems in pddl format using STRIPS syntax. You must have to write complete code so that I can actually run it with classical Planner and get the final plan in output. Ensure that all objects, predicates, initial conditions, and goal states are defined in complete detail. Do not leave any part of the code incomplete or abbreviated. Avoid placeholders or ellipses ("..."). When generating code, offer explanations for code segments as necessary under specified format and maintain good coding practices.

Instructions:
- Given the **domain definition**, **images**, and **problem description**, generate the corresponding **problem.PDDL** file.
  ```
  <ANSWER>
  Explaination:
  [Logic and explaination behind generated problem.pddl]

  Problem PDDL:
  [Insert the problem.PDDL content here]

  </ANSWER>
  ```
- Ensure that the PDDL code is syntactically correct and formatted properly.
- Ensure the plan and answer are clearly separated and labeled as shown.
- Under **Explaination**, include explanations, additional text, logic behind each step, and any code snippets.
- Under **Problem PDDL**, only include the problem PDDL code without any additional text and explainations. Dont add any extra backticks for code.

In-Context Examples

Question: *This is maze having 11 * 9 cells. The empty cells... What is the combined number of left and right turns do you need to make in this optimal path?*

images : [init and goal state images...]

Domain PDDL : [(define (domain maze)...]

Answer:

Problem PDDL: [content of problem.pddl]

Input Question

Question: *This is maze having 11 * 13 cells. The empty cells are coloured white and the obstacle cells are coloured black. From an empty cell, you can only move up, down, left, or right to another adjacent empty cell. You cannot move diagonally between two empty cells and cannot step into a cell with an obstacle. The entry cell of the maze is shown with the green arrow. The exit cell of the maze is shown with the blue arrow. Suppose you have found the most optimal path in the maze between the entrance and exit, where you need to go through the least number of empty cells and you need to make the least number of left and right turns. What is the total number of right turns do you need to make in this optimal path?*

images :

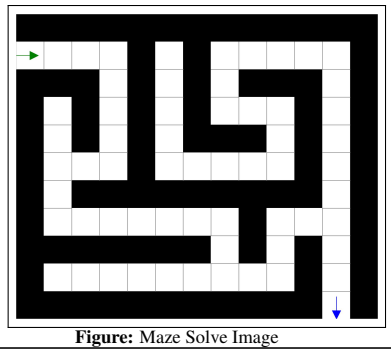


Figure: Maze Solve Image

Domain PDDL : [(define (domain maze)...]

Output

Explaination: *The problem defines a maze where an agent needs to navigate from an entry point to an exit point. The description provides the layout of the maze using visual representation. The domain PDDL file is provided, and it defines actions such as moving up, down, left, and right. We need to generate a problem PDDL file based on the given maze configuration, ensuring the initial conditions and goals represent the maze accurately.*

Problem PDDL:

```
(define (problem maze-problem)
  (:domain maze)
  (:objects
    ...
  )
  (:init
    ...
  )
  (:goal
    ...
  )
)
```

Table 9: PG using GPT4o in 1-shot incontext setting for Maze Solve puzzle

Instruction

```
You are an expert assistant proficient in solving planning problems and generating detailed solutions in natural language (NL).
Your task is to analyze a given problem described by a combination of:
1. A textual description of the puzzle.
2. Associated images representing the initial and goal states.
3. A domain definition and a problem PDDL file, which may be partially or fully correct, generated by an LLM in previous steps.

**Objective**:
- Generate a comprehensive NL solution plan and a clear final answer to the problem based on the inputs.

**Instructions**:
- Provide your response in the following format:
```
<ANSWER>

Detailed Plan:
[Comprehensive explanation of the plan, including the logical reasoning for each step, the goal, and the analysis of any potential
issues with the given PDDL file.]

Step Plan:
[Concise step-by-step instructions derived from the solution.]

Final Answer:
[The final conclusion or solution in natural language.]

</ANSWER>
```
- Ensure that **Detailed Plan** explains the logic behind each step and includes reasoning for any corrections or assumptions made
about the PDDL file.
- The **Step Plan** provides an actionable, step-by-step sequence to achieve the goal.
- The **Final Answer** offers a concise yet clear resolution to the problem. Ensure it will be a single integer denoting final
answer without any additional text.
- If there are inconsistencies or missing details in the problem PDDL, make reasonable assumptions, explain them in the Detailed
Plan, and proceed with generating the solution. Reference the domain and problem files explicitly when describing your reasoning
to ensure traceability. Use the images and description for additional context and validation of the plan where necessary.
```

Input Question

***Question**: You are given 3 jugs of capacities 13, 12, 6 litres. Initially, the amount of water that is contained in each jar is shown in the image. A single step of water pouring from one jug to another is constrained by the following rules: i) take a non-empty jug and pour water from it to another non-full jug until the first one becomes empty or the second one becomes full; and ii) no water can be spilt while pouring. The objective is to reach the amounts of 13, 12, 2 litres of water in the jugs from left to right, respectively. What is the minimum number of water pouring steps required to achieve the objective?*

images :

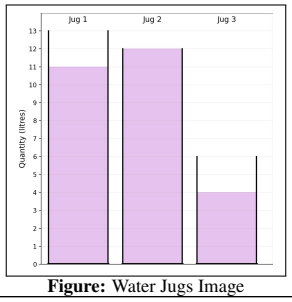


Figure: Water Jugs Image

Domain PDDL : [(define (domain waterjug)...]

***Problem PDDL**:* [(define (problem water-jug-pouring)...]

Output

***Detailed Plan**: The problem involves transferring water between three jugs to achieve a desired quantity of water in each jug. The jugs have capacities of 13, 12, and 6 litres respectively, and the initial state has 11 litres in Jug 1, 12 litres in Jug 2, and 3 litres in Jug 3. The goal state is to have 13 litres in Jug 1, 12 litres in Jug 2, and 2 litres in Jug 3. Let's first analyze the initial state and the goal state and determine the steps needed to reach the goal:*

1. Jug 1 contains 11 litres, 2. Jug 2 contains 12 litres, 3. Jug 3 contains 4 litres.

The objective is to have: 1. Jug 1 containing 13 litres, 2. Jug 2 containing 12 litres, 3. Jug 3 containing 2 litres.

To meet this objective, some water needs to be redistributed between the jugs. The primary goal is to transfer water from Jug 3 to Jug 1 or Jug 2 such that Jug 3 ends up with only 2 litres of water. Here is the step-by-step plan:

1. Pour water from Jug 3 into Jug 1 until Jug 1 is full.

- Jug 1: 11 + 2 = 13 litres (full)

- Jug 2: 12 litres (unchanged)

- Jug 3: 4 - 2 = 1 litre (since Jug 1 only needed 2 litres to become full):

In simpler terms, the idea is to ensure that Jug 3 ends up having exactly 2 litres by working its content down through specific transfer steps.

***Step Plan**:*

1. Pour 2 litres from Jug 3 into Jug 1.

***Final Answer**: 1*

Table 10: Prompt for SG (GPT4o): Plan generation from the output of PG in 1-shot in-domain setting for the Water Jugs puzzle

Instruction

```
**Instructions**:
- Provide your response in the following format:
Answer: Let's describe the image first and think step by step. Strictly, provide the final answer
option between [(A), (B), (C), (D)]
```
```
<ANSWER>
Explanation:
[Logic and explaination behind solving puzzle and generating plan]

Option:
[The single correct option from [(A), (B), (C), (D)] that matches the derived answer. Don't write any additional text or
integer answer. When you could not find an answer from the given options, write NONE only without any explaination.]
</ANSWER>
```
```

**Input Question**

***Question***: *Consider a sliding block puzzle of grid size 5 * 4 units. It has 9 wooden blocks of varying sizes: one 2 * 2, four 1 * 2, two 2 * 1, and two 1 * 1. The gird also has two empty 1 * 1 spaces. The blocks cannot be removed from the grid, and may only be slid horizontally and vertically within its boundary. A move is defined as selecting a block that is slideable, and moving it by 1 unit either horizontally or vertically, whichever is possible. The images show the starting and ending configurations of the puzzle grid. The wooden blocks are shown in various shades of brown and the empty spaces are shown in white. What is the minimum number of moves required to reach the ending configuration from the starting configuration? Do not directly give the final answer.*

***Options***:*['A': 4, 'B': 5, 'C': 3, 'D': 2]*

***images :***

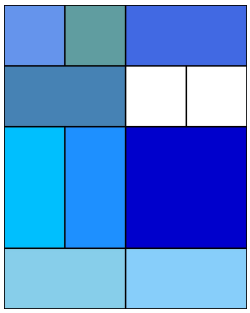


**Figure:** Initial Configuration

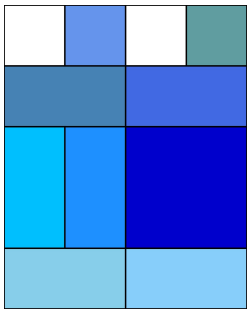


**Figure:** Goal Configuration

**Output**

***Explanation***:

*1. Move the top right 1x1 block left by 1 unit into one of the empty spaces.*

*2. Slide the 1x2 block that was just cleared down by 1 unit into the other empty space.*

*3. Slide the other 1x2 block right by 1 unit into the space created by the previous move.*

*4. Finally, move the 1x1 block currently in the middle left by 1 unit.*

***Option***: *(A)*

Table 11: eCOT (multi-choice) using GPT4o for wood slide puzzle
```