PUZZLEPLEX: Benchmarking Foundation Models on Reasoning and Planning with Puzzles

Anonymous ACL submission

Abstract

This work investigates the reasoning and planning capabilities of foundation models and their scalability in complex, dynamic environments. We introduce PUZZLEPLEX, a benchmark designed to assess these capabilities through a diverse set of puzzles. PUZZLEPLEX consists of 15 types of puzzles, including deterministic and stochastic games of varying difficulty, as well as single-player and two-player scenarios. The PUZZLEPLEX framework provides a comprehensive environment for each game, and supports extensibility to generate more challenging instances as foundation models evolve. Additionally, we implement customized game-playing strategies for comparison. Building on this benchmark, we develop fine-grained metrics to measure performance and conduct an in-depth analysis of frontier foundation models across two settings: *instruction-based* and *code-based*. Furthermore, we systematically investigate their scaling limits. Our findings show that reasoning models outperform others in instruction-based settings, while code-based execution presents greater challenges but offers a scalable and efficient alternative. PUZZLEPLEX enables targeted evaluation and guides future improvements in reasoning, planning, and generalization for foundation models. [1]

1 Introduction

The rapid progress of foundation models has led to remarkable improvements across a broad spectrum of natural language processing tasks. Recently, the emergence of reasoning models such as OpenAI o-series models (OpenAI et al., 2024a) and DeepSeek-R1 (DeepSeek-AI et al., 2025a) have demonstrated remarkable advances in complex reasoning tasks through test-time compute scaling. These breakthroughs naturally prompt a deeper question: *How far can modern models push genuine problem-solving ability, especially in scenarios that demand sustained, structured reasoning?*

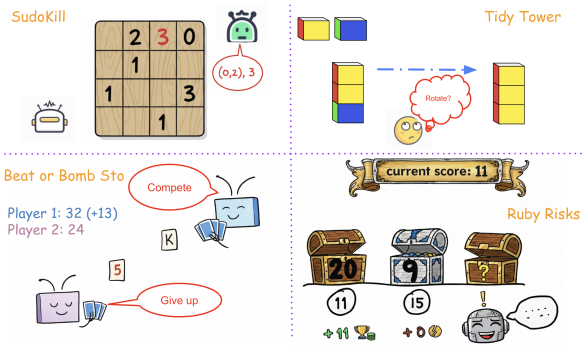

Figure 1: Overview of four puzzles: **SudoKill** (two-player deterministic), **Tidy Tower** (single-player deterministic), **Beat or Bomb Sto** (two-player stochastic), and **Ruby Risks** (single-player stochastic).

To explore this, we turn to puzzle solving—a domain that inherently blends logical, numerical, and spatial reasoning with long-horizon planning and strategic adaptation. Many puzzles unfold over multiple interactive steps, involving competition or dynamic environments. This makes them ideal for evaluating a model's ability to reason under evolving constraints, adapt to new strategies, and maintain coherence across extended interactions.

To this end, we introduce PUZZLEPLEX, a benchmark designed to evaluate foundation models' reasoning and planning capabilities. Unlike prior benchmarks (Zhang et al., 2025; Wu et al., 2024b) that reuse common puzzles—many potentially seen during pretraining—PUZZLEPLEX features 15 novel, curated puzzles spanning both **text-only** and **text-image** formats. As shown in Figure 1, the puzzles cover **single-player** and **two-player** settings and include both **deterministic** and **stochastic** environments. Each puzzle supports multiple difficulty levels and extensible generation, enabling adaptive evaluation as models improve. Their long-horizon, dynamic nature pro-

[1] The code and data are available on https://anonymous.4open.science/r/PuzzlePlex_dataset-6F20.

Benchmark	Game Scenario		Reward Predictability		# Multi-	Data Type		Varying	Evaluation	
	Single-	Two-player	Deterministic	Stochastic	Turn	Text	Text-Image	Difficulty	Inst.	Code
PUZZLEBENCH (Mittal et al., 2025)	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓
GRIDPUZZLE (Tyagi et al., 2024)	✓	✗	✓	✗	✗	✓	✗	✓	✓	✗
ZEBRALOGIC (Lin et al., 2025)	✓	✗	✓	✗	✗	✓	✗	✓	✓	✗
MASTERMINDEVAL (Golde et al., 2025)	✓	✗	✓	✗	✓	✓	✗	✓	✓	✗
PUZZLES (Estermann et al., 2024)	✓	✗	✓	✗	✓	✗	✓	✓	✗	✓
LOGICGAME (Gui et al., 2024a)	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗
BOARDGAMEQA (Kazemi et al., 2023)	✓	✗	✓	✓	✗	✓	✗	✗	✓	✗
PUZZLEQA (Zhao and Anderson, 2023)	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗
ENIGMAEVAL (Wang et al., 2025)	✓	✗	✓	✗	✓	✓	✓	✓	✓	✗
VGRP-BENCH (Ren et al., 2025)	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗
SMARTPLAY (Wu et al., 2024a)	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗
PUZZLEPLEX (ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison between PUZZLEPLEX and existing puzzle benchmarks. A single-turn game ends after one move by one or more players.

vides a compact yet demanding testbed for assessing reasoning depth, planning, and strategic coherence—areas underexplored in prior short-context benchmarks (Gui et al., 2024b).

We further design hand-crafted strategies for comparison and evaluate models under two complementary paradigms: **instruction-based** and **code-based**. In the former, models act as agents interacting via natural language; in the latter, they generate executable code that solves the puzzle. Together, these paradigms reveal both interactive reasoning and programmatic abstraction capabilities.

Empirical results show that reasoning models outperform non-reasoning ones in instruction-based settings, leveraging test-time scaling and extended deliberation. However, performance drops in code-based evaluation due to challenges in program synthesis, though sampling-based methods help narrow the gap. Open-source models increasingly rival proprietary systems, and visual or legality-aware prompting further boosts results. However, models still struggle with multi-hop reasoning in some puzzles, suggesting limitations in their ability to maintain coherent reasoning over extended contexts.

In summary, our contributions are:

- PUZZLEPLEX, the first benchmark to jointly evaluate reasoning in both interactive and executable settings across diverse puzzle types.
- Framework that supports textual and visual puzzles with deterministic and stochastic dynamics.
- Hand-crafted baselines and fine-grained metrics enabling rigorous evaluation and comparison of systems and reasoning strategies.
- Comprehensive empirical analysis across leading models, comparing the performance of different reasoning strategies, the scaling behavior of different systems, and the systems' failure modes.

2 Related Work

2.1 Puzzles and Relevant Benchmarks

Puzzles can be broadly divided into **rule-based** and **rule-less** types. Rule-based puzzles, such as SUDOKU (Noever and Burdick, 2021), CROSSWORDS (Sadallah et al., 2025), and CHESS (Feng et al., 2023), have explicit rules, defined goals, and structured state transitions, requiring strategic and logical reasoning. Rule-less puzzles, including Riddles (Lin et al., 2021; Bisk et al., 2019), lack explicit action spaces or clear objectives. PUZZLEPLEX focuses on **rule-based** puzzles to enable objective evaluation of reasoning in competitive, dynamic settings. We exclude knowledge-heavy puzzles (e.g., GUESS MY CITY (Abdulhai et al., 2023)) that depend on external knowledge (Schuster et al., 2021; Lin et al., 2021; Todd et al., 2024), since modern models already trained extensively on factual corpora and outperform humans on such tasks.

Table 1 compares PUZZLEPLEX with recent puzzle benchmarks. Most existing benchmarks focus on **single-player**, short-horizon puzzles (Mittal et al., 2025; Gui et al., 2024a; Zhao and Anderson, 2023), while multi-turn, competitive two-player settings are rarely explored (Wu et al., 2024a; Liu et al., 2023). Few benchmarks incorporate **stochastic environments** for reasoning under uncertainty, or **multimodal puzzles** that require joint text-image understanding (Estermann et al., 2024; Wang et al., 2025; Ren et al., 2025).

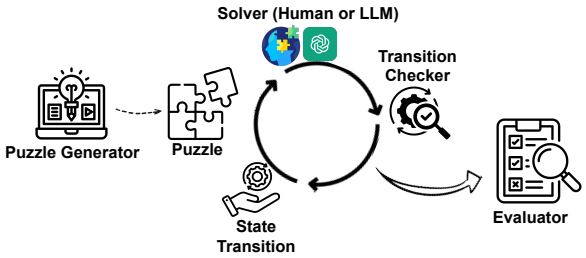

Figure 2: Overview of the developed pipeline framework. **Puzzle Generator** creates puzzle instances from templates based on the puzzle name, difficulty level, and selected competing models. The **Solver** then generates a response after receiving the puzzle instance. This response is passed to the **Transition Checker**, which verifies the legality of the operation output by the **Solver** and checks the game status. If the game ends, the **Evaluator** calculates and outputs the score. Otherwise, **State Transition** updates the state and passes the updated information back to the **Solver**.

2.2 Evolution of Puzzle Solving Techniques

A variety of methods have been developed for solving rule-based puzzles. Classical approaches rely on algorithmic techniques such as dynamic programming (Smith, 2007), alpha–beta pruning (Korf, 1990), and heuristic search (Lewis, 2007). For single-player puzzles, neurosymbolic methods (Ahmed et al., 2023; Murali et al., 2022) are effective due to their combinatorial nature, often reducible to SAT or SMT formulations (Bright et al., 2020; Høfler, 2014).

With deep learning advances, reinforcement learning (RL) has become the dominant paradigm (dos Santos et al., 2019; Huang et al., 2024), though combinatorial explosion still necessitates heuristics (Silver et al., 2016). Early model-based approaches fine-tuned models like GPT-2 (Radford et al., 2019) and FLAN-PaLM (Chung et al., 2022) for puzzles such as Sudoku (Noever and Burdick, 2021) and BoardgameQA (Kazemi et al., 2023). Stronger foundation models (OpenAI et al., 2024b; Anthropic, 2024) now solve puzzles through few-shot in-context learning and multi-run feedback. Among prompting methods, Chain-of-Thought (CoT) (Wei et al., 2023) consistently outperforms direct prompting, while extensions like Self-Refine (Madaan et al., 2023), Tree-of-Thought (ToT) (Yao et al., 2023), and Everything-of-Thoughts (Ding et al., 2024) further enhance reasoning for deterministic puzzles.

In this work, we adopt CoT-style prompting to evaluate systematic reasoning, planning, and decision-making. We further introduce a **code-based execution** setting, where models generate and execute code to interact directly with puzzle environments—linking reasoning with concrete actions and improving solution correctness and generalization.

3 PUZZLEPLEX

We first introduce the PUZZLEPLEX framework in which puzzle templates can be instantiated, moves recorded, state information shared, and states evaluated. We next describe the puzzles included in this benchmark, the implementation of customized strategies, and the evaluation methods.

3.1 Puzzle Generation Framework

PUZZLEPLEX has the following main components, as presented in Figure 2.

Instance Generation. For each puzzle p, we distinguish between a possibly parametrized puzzle template $template(p)$ (e.g., SudoKill on a 9 $\times$ 9 grid, template(SudoKill(9,9)), and an instance $instance(p)$ (e.g., a particular instance of SudoKill on a 9 $\times$ 9 grid, instance(Suduoku(9,9)). A generator function G_p maps templates to instances. The generated instance is also the initial state S_0 of the game. That is, $instance(p) = S_0$. The generator for each puzzle will create instances using randomness, and it will adjust the difficulty level by varying the size of the puzzle.

State Transition. After receiving a move M generated by a player (human or computer), the state transition module maps a state S_n to a new state S_{n+1} while incorporating feedback F_n. The feedback F_n indicates the legality of the move, whether the game has terminated, and provides new position information. This process is represented as $M : S_n \to (S_{n+1}, F_n)$.

Evaluation. Once the puzzle-solving process terminates, an **Evaluator** E_p is applied to the sequence of states $S_0, S_1, \ldots, S_n$ to determine the raw score(s), represented as $rs_p = E_p(S_0, S_1, \ldots, S_n)$. The scale of the raw scores varies depending on the resolution type of each puzzle. To ensure comparability, we normalize these scores to obtain final scores ranging from 0 to 1 (§ 3.5).

To better keep track of state transitions and model reasoning steps, we implemented a Web UI called **Simulator** for visual observation. An example of this interface is shown in the § A.2.

Model	Single-Player Det.		Two-Player Det.		Score
	Easy	Normal	Easy	Normal	
Custom	0.89 ±0.47	0.83 ±0.74	0.59 ±0.36	0.60 ±0.34	0.70 ±0.15
Deepseek-R1	0.64 ±1.38	0.48 ±1.04	0.66 ±0.12	0.66 ±0.14	0.62 ±0.15
o4-mini	0.44 ±1.16	0.44 ±1.14	0.67 ±0.15	0.68 ±0.12	0.59 ±0.15
Gemini-2.5-pro	0.44 ±1.13	0.44 ±1.02	0.68 ±0.14	0.67 ±0.13	0.58 ±0.14
QwQ-32B	0.54 ±1.15	0.25 ±0.58	0.69 ±0.10	0.68 ±0.11	0.58 ±0.14
grok-3-mini	0.17 ±0.52	0.22 ±0.51	0.67 ±0.20	0.67 ±0.20	0.49 ±0.15
Deepseek-V3	0.34 ±0.78	0.24 ±0.54	0.52 ±0.13	0.50 ±0.14	0.43 ±0.10
GPT-4.1	0.40 ±0.98	0.35 ±0.94	0.44 ±0.13	0.45 ±0.09	0.42 ±0.11
Qwen-2.5-VL-72B	0.24 ±0.62	0.24 ±0.37	0.24 ±0.18	0.28 ±0.24	0.25 ±0.09
Llama-3.3-70B	0.15 ±0.37	0.12 ±0.39	0.31 ±0.12	0.31 ±0.12	0.25 ±0.07
Gemma-3-27B	0.13 ±0.38	0.12 ±0.39	0.23 ±0.12	0.24 ±0.10	0.19 ±0.06
Phi-4-multimodal	0.05 ±0.20	0.03 ±0.14	0.17 ±0.12	0.17 ±0.05	0.12 ±0.05

Table 2: Instruction-based normalized scores (mean ± 95% CI) of models on single-player and two-player deterministic puzzles, separated by difficulty.

3.2 PUZZLEPLEX Benchmark Construction

All puzzles in PUZZLEPLEX are either derived from a column in Communications of the ACM [2] or manually curated by the authors. While foundation models may have been exposed to textual descriptions of these puzzles, there are no publicly available strategies for solving them, thereby minimizing the risk of data contamination during gameplay. Additionally, we have simplified the rules of several puzzles to reduce the barrier to entry, enabling most users to engage with them immediately after learning the rules and objectives.

Our 15 puzzles are categorized into four types: **single-player deterministic**, **single-player stochastic**, **two-player deterministic**, and **two-player stochastic**. Text-based puzzles span all four types, whereas text-image puzzles are limited to the two-player deterministic type. The distinction between deterministic and stochastic games lies in the predictability of operation outcomes. In deterministic games, the result of a decision is fixed, regardless of how many times it is taken. In contrast, stochastic games produce probabilistic outcomes, where repeated execution of the same operation in the same state may lead to different results. We choose these puzzles because they involve: (i) constraint satisfaction, (ii) sequential decision-making over game states, (iii) adversarial two-player interaction, and (iv) stochastic dynamics. Many instances admit CSP-style formulations which are NP-hard in general (Garey and Johnson, 2002). By covering diverse mechanics and controlled difficulty, we reduce shortcut solutions and better isolate rule generalization and strategic reasoning. Detailed information about the puzzles is provided in §A.1, and individual puzzle descriptions are included in §A.3.

[2] https://cacm.acm.org/section/opinion/

3.3 Customized Strategies

We implemented customized strategies for each puzzle, which can be categorized as follows:

- **Brute-force Algorithm**: This method is employed when the problem size allows for an exhaustive search within our specified time constraints.
- **Search Algorithms**: We employ a variety of search techniques, including both uninformed and probabilistic methods. Specifically, we use Breadth-First Search (BFS) and Depth-First Search (DFS) as examples of uninformed search strategies. Monte Carlo Tree Search (MCTS) is incorporated as a form of probabilistic search.
- **Dynamic Programming (DP)**: Dynamic programming is applied to puzzles that exhibit overlapping subproblems and optimal substructure.
- **Greedy Algorithm**: Greedy algorithms are employed in puzzles where locally optimal choices are expected to lead to globally optimal solutions or the search space is too large for other techniques, often reflecting strategies used in real-world scenarios.
- **Other Methods**: These include other algorithms, such as backtracking and simulated annealing.

3.4 Evaluation Protocols

To gain a holistic view of a model's problem-solving ability under distinct modes of interaction, we design the following two evaluation protocols.

Model	Single-Player Det.				Single-Player Sto.				Two-Player Det.				Two-Player Sto.				Score	
	Easy		Normal		Easy		Normal		Easy		Normal		Easy		Normal		Avg.	Best
	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best		
Custom	0.89	–	0.83	–	0.75	–	0.80	–	0.59	–	0.75	–	0.55	–	0.72	–	0.73	–
Deepseek-R1	0.33	0.53	0.25	0.43	0.54	0.89	0.51	0.94	0.65	0.77	0.66	0.82	0.46	0.52	0.53	0.61	0.52	0.66
o4-mini	0.30	0.42	0.32	0.52	0.44	0.89	0.48	0.93	0.69	0.85	0.73	0.85	0.49	0.52	0.57	0.60	0.53	0.73
Gemini-2.5-pro	0.34	0.68	0.31	0.65	0.36	0.69	0.36	0.76	0.66	0.82	0.74	0.94	0.47	0.52	0.47	0.54	0.50	0.74
QwQ-32B	0.21	0.42	0.07	0.20	0.48	0.85	0.31	0.81	0.59	0.68	0.34	0.58	0.53	0.65	0.37	0.54	0.37	0.59
grok-3-mini	0.28	0.46	0.20	0.38	0.22	0.64	0.12	0.65	0.59	0.75	0.68	0.78	0.54	0.71	0.62	0.76	0.43	0.65
Deepseek-V3	0.22	0.40	0.16	0.34	0.54	0.86	0.35	0.93	0.51	0.64	0.45	0.60	0.45	0.52	0.40	0.54	0.40	0.54
GPT-4.1	0.24	0.43	0.28	0.48	0.44	0.88	0.44	0.93	0.63	0.77	0.72	0.81	0.56	0.73	0.65	0.81	0.51	0.72
Qwen-2.5-VL-72B	0.19	0.36	0.16	0.49	0.41	0.80	0.30	0.86	0.52	0.69	0.55	0.68	0.45	0.70	0.46	0.73	0.40	0.55
Llama-3.3-70B	0.18	0.41	0.17	0.40	0.40	0.82	0.30	0.84	0.50	0.69	0.56	0.62	0.50	0.66	0.60	0.64	0.41	0.60
Gemma-3-27B	0.22	0.43	0.22	0.41	0.35	0.79	0.32	0.82	0.45	0.59	0.51	0.61	0.46	0.66	0.47	0.65	0.38	0.51
Phi-4-multimodal	0.00	0.00	0.00	0.00	0.06	0.24	0.01	0.20	0.05	0.07	0.05	0.07	0.19	0.25	0.05	0.08	0.05	0.19

Table 3: Code-based normalized scores.

Instruction-based Evaluation. Single-player deterministic puzzles are evaluated using 10 randomly generated instances with fixed seeds from 1 to 10 to ensure reproducibility. For two-player deterministic puzzles, each model pair competes on 5 instances (seeds 1–5), with each match repeated twice while alternating the first player to account for first-mover advantage. All evaluations are conducted at two difficulty levels: *easy* and *normal*. Stochastic puzzles are excluded from this setting due to their inherent variance and the high cost of running enough instances to achieve statistically robust conclusions.

Code-based Evaluation. Code-based setting requires the model to produce an *executable* policy, providing a stricter test of abstract reasoning and generalization. Concretely, it demands **(1) rule formalization**—translating natural-language rules and edge cases into precise state updates and legality checks; **(2) constraint-aware strategy under intractability**—many puzzles resemble CSP/search problems where brute force is impractical, so effective solutions rely on pruning and heuristics; and **(3) protocol compliance for automatic evaluation**—adhering to a fixed I/O and interaction format so rollouts are reproducible and machine-checkable. Accordingly, we track both task performance and execution outcomes (e.g., *syntax/runtime errors* and *timeouts*), which directly reflect failures in robust rule understanding and algorithmic control. In this setting, each foundation model is sampled 32 times per puzzle to generate code, following the prompt templates described in §B.7. The resulting programs are then executed to play the games. For deterministic puzzles, we follow the same evaluation protocol as in the instruction-based setting. For single-player stochastic puzzles, each generated program is evaluated over 100 runs (seeds 1 to 100) across both difficulty levels. For two-player stochastic puzzles, each program competes in 50 runs (seeds 1 to 50), alternating player roles in each match.

3.5 Evaluation Metrics

We employ two primary metrics to evaluate model performance: **Normalized Score** and **Elo Score**, both derived from raw scores.

Raw Score. In single-player games, raw scores are either binary or continuous. Binary puzzles assign a score of 1 for success and 0 for failure. Continuous-score puzzles assign values based on criteria such as move count, constraints met, or objectives achieved, and scores may fall outside the [0, 1] range. In two-player games, outcomes are categorized as win, loss, or tie, corresponding to scores of 1, 0, and 0.5, respectively.

Normalized Score. For two-player games, raw scores already lie in [0, 1] and do not require normalization. For single-player games, normalization ensures comparability by rescaling scores to the [0, 1] interval. This involves determining the best and worst achievable scores under identical initialization conditions. If higher scores are better, the top-performing model is assigned 1 and others receive $score/max$; if lower is better, normalization uses $min/score$.

Elo Score. To enable unified comparison across both single-player and two-player settings, we apply the Elo rating system, a widely-used model comparison metric (Boubdir et al., 2023). For single-player games, we create pairwise matchups between models based on their normalized scores—the model with the higher normalized score is considered the winner in each pairwise

comparison. The implementation details are described in § B.4.

4 Experiments

4.1 Experimental Setup

The foundation models we evaluate include GPT-4.1* (OpenAI, 2025a), o4-mini* (OpenAI, 2025b), Gemini-2.5-pro* (Google, 2025), grok-3-mini* (xAI, 2025), DeepSeek-V3 (DeepSeek-AI et al., 2025b), DeepSeek-R1 (DeepSeek-AI et al., 2025a), QwQ-32B (Qwen, 2024), Qwen-2.5-VL-72B (Bai et al., 2025), Gemma-3-27B (Team et al., 2025), Llama-3.3-70B (Grattafiori et al., 2024), and Phi-4-multimodal (Microsoft et al., 2025).[3] Models grok-3-mini, DeepSeek-V3, DeepSeek-R1, and QwQ-32B do not support image modalities and are therefore excluded from evaluation on text-image puzzles in the instruction-based setting. We use the chat or instruct versions of each model, as solving most puzzles involves multi-turn interactions.

4.2 Main Results

Table 2 presents the normalized scores of all models under the instruction-based setting, while Elo scores are reported in Table 9 in Appendix. A breakdown of scores for each puzzle is provided in § C.1. For the code-based setting, results are shown in Table 3.

Reasoning models outperform non-reasoning models in the instruction-based setting. From Table 2, we observe that reasoning models consistently outperform non-reasoning ones, with all top-5 models employing reasoning strategies. This demonstrates the effectiveness of test-time scaling using extended CoT, where deeper deliberation translates to better performance in gameplay. Notably, the relatively small QwQ-32B model surpasses larger non-reasoning models such as GPT-4.1 and DeepSeek-V3. Furthermore, open-source models are highly competitive with proprietary systems: for instance, DeepSeek-R1 achieves the highest normalized score of 0.62, outperforming Gemini-2.5-pro, the best-performing proprietary model, which scores 0.58. These findings indicate that open-source models are closing the performance gap. Although foundation models still lag behind our custom strategy (which scores 0.70) on average, several leading models perform comparably—or even better—in two-player deterministic puzzles, highlighting the rapid progress of foundation models.

Code-based setting is more challenging and leads to a notable performance drop. As shown in Table 3, model performance declines significantly in the code-based setting, where models must generate executable code to play the games autonomously. Unlike the instruction-based setting—where models act as interactive agents with ongoing access to game states and can adjust actions dynamically—the code-based setting demands strong program synthesis capabilities. This shift reduces the advantage of reasoning models: for example, GPT-4.1, a non-reasoning model, ranks among the top-3 performers in the code-based setting, whereas no non-reasoning model appears in the top-5 for the instruction-based setting.

The performance drop is especially evident in single-player deterministic puzzles. DeepSeek-R1, for instance, sees its score decline from 0.64 to 0.33 in the easy level, and from 0.48 to 0.25 in the normal level. Table 16 in Appendix further reveals a significant reduction in win rates against the customized strategy across all models in the code-based setting in two-player deterministic games. Although the code-based setting underperforms compared to the instruction-based setting, its lower computational cost makes it a promising direction.

These results underscore the greater difficulty of the code-based setting, which not only tests reasoning but also code generation and execution accuracy. However, one advantage of this setting is efficiency: code is generated once per puzzle and can be reused. In our experiments, each model generates 32 samples per puzzle. As shown in Table 3, the best scores from code-based runs can approach or even match the performance of the customized strategy.

4.3 More Instruction-based Analysis

Mixed Effectiveness of Advanced Prompting Strategies. Table 12 in Appendix reports GPT-4.1 and o4-mini results on TIDYTOWER and SUDOKILL across prompting strategies. Overall, advanced prompting yields mixed gains: 1-shot provides little to no improvement, and ToT helps on TIDYTOWER but offers minimal benefit on SUDOKILL. Notably, prompting *without history* substantially boosts TIDYTOWER, outperforming ToT despite ToT's much higher compute cost, suggest-

[3] Models marked with an asterisk (*) are proprietary.

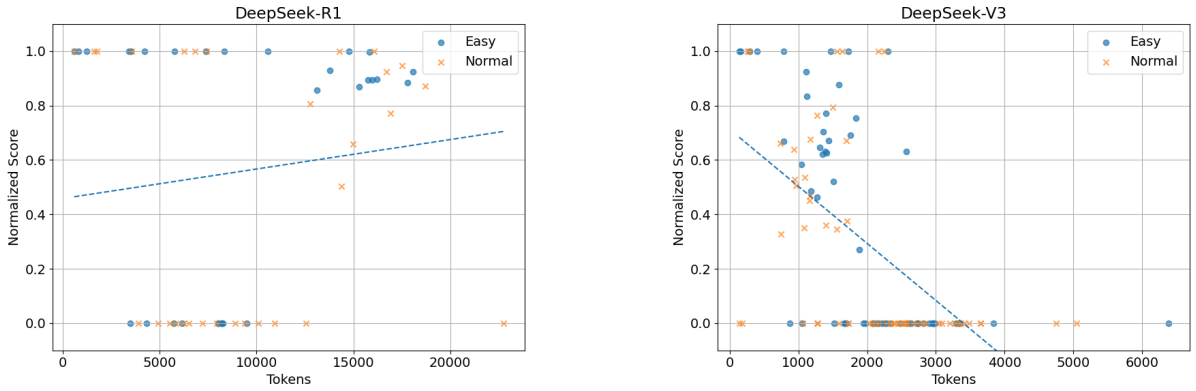

Figure 3: Comparison between the reasoning model Deepseek-R1 and the non-reasoning model Deepseek-V3 in terms of generated token counts versus normalized scores on single-player deterministic puzzles.

ing that current models still struggle with multi-hop reasoning and that including prior reasoning traces can sometimes mislead rather than help. We also test a **legality-aware** strategy that supplies a list of legal candidate moves, motivated by frequent failures due to illegal actions causing immediate losses or premature termination; this consistently improves performance in Table 12. Finally, o4-mini benefits more from these prompting strategies than GPT-4.1.

Evaluating Multimodal Integration in Strategic Reasoning. Table 13 shows that most models benefit from incorporating visual inputs, confirming the value of image-based state representations in puzzle-solving tasks. High-capacity models like o4-mini and GPT-4.1 achieve notable gains, with GPT-4.1 improving its win rate by +0.38 on SUPERPLYM (Normal). However, weaker models such as Phi-4-multimodal struggle to utilize visual information effectively, sometimes exhibiting performance drops (e.g., -0.75 on SUPERPLYM Easy). These results suggest that while visual information aids intuitive understanding, effective multimodal reasoning requires advanced fusion capabilities. The benefits are more pronounced in simpler tasks, whereas complex scenarios demand stronger cross-modal reasoning, which current models often lack.

4.4 Scaling Analysis

Reasoning models demonstrate better scaling between token count and performance. Figure 3 compares Deepseek-R1 (reasoning) and Deepseek-V3 (non-reasoning) in terms of total generated tokens (reasoning + completion) versus normalized scores on single-player deterministic puzzles. These puzzles are all single-pass and do not involve multi-round interactions, making them suitable for such analysis. The results show that for Deepseek-R1, performance generally improves with increased token generation, suggesting effective test-time scaling. In contrast, Deepseek-V3 exhibits a flatter or even downward trend, indicating limited benefit from generating more tokens. Furthermore, Deepseek-R1 tends to allocate more tokens to normal-difficulty instances than to easy ones, aligning with task complexity, while Deepseek-V3 shows little variation across difficulty levels.

Reasoning models show improved performance in instruction-based settings but mixed results in code generation. For each run instance, we define several termination statuses. LEGAL means the game ends normally. RULE VIOLATION occurs when models make moves that violate the rules, causing game termination. NOT FOLLOWING INSTRUCTION indicates that foundation models fail to follow instructions properly; in instruction-based settings, this means the model generates data in a format that prevents the system from extracting moves; in code-based settings, it means the model fails to generate code meeting our requirements. TIMEOUT is a status exclusive to code-based settings, indicating that the model-generated code exceeds our predetermined runtime limit, forcing the game to stop. SYNTAX ERROR, also specific to code-based settings, occurs when the model generates code containing syntax errors. RUNTIME ERROR, another code-based status, happens when code executes but fails during runtime due to errors such as index exceptions.

From Table 4, which shows the distribution of status types and average token usage per model

Model	Status Type						#Token
	Legal	Not Follow Instr.	Timeout	Rule Violation	Runtime Err.	Syntax Err.	
Instruction-based							
Deepseek-R1	0.79	0.01	–	0.20	–	–	9420.78 ±710.39
o4-mini	0.79	0.02	–	0.19	–	–	4508.83 ±557.87
Gemini-2.5-pro	0.72	0.17	–	0.12	–	–	12124.58 ±1016.97
QwQ-32B	0.78	0.01	–	0.21	–	–	11840.75 ±835.78
grok-3-mini	0.82	0.04	–	0.15	–	–	12479.65 ±976.22
Deepseek-V3	0.72	0.03	–	0.25	–	–	2013.47 ±95.00
GPT-4.1	0.67	0.01	–	0.32	–	–	1587.50 ±137.30
Qwen-2.5-VL-72B	0.57	0.05	–	0.38	–	–	654.02 ±50.49
Llama-3.3-7B	0.61	0.00	–	0.38	–	–	949.28 ±74.05
Gemma-3-27B	0.37	0.44	–	0.19	–	–	1236.40 ±79.88
Phi-4-multimodal	0.42	0.17	–	0.41	–	–	765.48 ±87.38
Code-based							
Deepseek-R1	0.54	0.18	0.01	0.00	0.11	0.16	11977.00 ±7694.16
o4-mini	0.61	0.26	0.03	0.03	0.05	0.03	1870.17 ±1379.59
Gemini-2.5-pro	0.58	0.13	0.02	0.14	0.13	0.00	14821.33 ±10064.59
QwQ-32B	0.19	0.07	0.00	0.01	0.19	0.54	10742.93 ±7226.60
grok-3-mini	0.56	0.19	0.03	0.05	0.13	0.04	14708.74 ±12639.34
Deepseek-V3	0.26	0.61	0.00	0.03	0.07	0.03	1133.52 ±1235.67
GPT-4.1	0.57	0.18	0.02	0.03	0.21	0.01	1287.01 ±1022.89
Qwen-2.5-VL-72B	0.46	0.13	–	0.09	0.04	0.27	607.89 ±286.81
Llama-3.3-7B	0.46	0.20	0.02	0.11	0.15	0.06	741.86 ±347.07
Gemma-3-27B	0.43	0.35	–	0.16	0.06	0.01	918.74 ±545.65
Phi-4-multimodal	0.00	0.50	–	0.00	0.03	0.48	587.53 ±656.44

Table 4: Distribution of status types and average tokens used per model in instruction-based and code-based settings.

in two different settings, we observe that in instruction-based settings, most reasoning models consume significantly more tokens than non-reasoning models, with typical reasoning models using more than five times the tokens of their non-reasoning counterparts (though o4-mini is an exception with more modest token usage). In code-based settings, o4-mini's token usage remains similar to instruction-based settings, while other reasoning models consume substantially more tokens—approximately ten times that of non-reasoning models.

Regarding status types in instruction-based settings, reasoning models generally make fewer errors, suggesting that increased reasoning tokens at test-time correlate with error reduction. However, in code-based settings, the situation differs. While existing research demonstrates that large reasoning models excel in competitive programming (OpenAI et al., 2025), our puzzle scenario yields different results. The table indicates that the best non-reasoning model, GPT-4.1, remains comparable to reasoning models, while one reasoning model, QwQ-32B, shows a notably low legal rate due to a high incidence of syntax errors in its code generation.

5 Conclusion

PUZZLEPLEX is the first benchmark to compare reasoning techniques on puzzles that span text and vision modalities, deterministic and stochastic dynamics, and long-horizon interactions. It enables a systematic evaluation of models through both instruction-based and code-based settings. We find that reasoning models perform best in instruction-based settings, benefiting from increased test-time computation. Open-source models such as DeepSeek-R1 match or surpass proprietary models, demonstrating rapid progress. In contrast, the code-based setting poses greater challenges due to the need for accurate program synthesis, though its lower computational cost and scalability make it a promising direction. Best-of-n sampling significantly improves performance in this setting. Multimodal inputs and legality-aware prompting offer further gains in specific scenarios. However, our analysis reveals that models often struggle with multi-hop reasoning—e.g., in TIDYTOWER, removing prior reasoning history improves accuracy, suggesting that current models may be misled by irrelevant context. Overall, PUZZLEPLEX offers a testbed for advancing reasoning and planning in foundation models, highlighting limitations of current systems and consequent opportunities for future research.

Limitations

Although PUZZLEPLEX spans 15 carefully curated puzzles, the overall number of puzzles remains modest, so results may be sensitive to the specific puzzle mix and random seeds. Moreover, due to rapid model evolution and budget constraints, our experiments may not include the latest model releases available after the experiment period. Finally, PUZZLEPLEX does not yet assess whether fine-tuned models can outperform existing LLMs, which could provide additional insights.

Ethics Statement

Our study uses only rule-based puzzles—no human subjects or personally identifiable information. All puzzles are original or permissively licensed. We execute model-generated code in a sandbox, follow provider terms/safety policies, and log only non-sensitive metadata.

References

Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. 2023. Lmrl gym: Benchmarks for multi-turn reinforcement learning with language models. *Preprint*, arXiv:2311.18232.

Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. 2023. Semantic strengthening of neuro-symbolic learning. *Preprint*, arXiv:2302.14207.

AI Anthropic. 2024. Introducing claude 3.5 sonnet.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Meriem Boubdir, Edward Kim, Beyza Ermis, Sara Hooker, and Marzieh Fadaee. 2023. Elo uncovered: Robustness and best practices in language model evaluation. In *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 339–352, Singapore. Association for Computational Linguistics.

Curtis Bright, Jürgen Gerhard, Ilias Kotsireas, and Vijay Ganesh. 2020. *Effective Problem Solving Using SAT Solvers*, page 205–219. Springer International Publishing.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025b. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Everything of thoughts: Defying the law of penrose triangle for thought generation. *Preprint*, arXiv:2311.04254.

Thiago Freitas dos Santos, Paulo E. Santos, Leonardo A. Ferreira, Reinaldo A. C. Bianchi, and Pedro Cabalar. 2019. Heuristics, answer set programming and markov decision process for solving a set of spatial puzzles. *Preprint*, arXiv:1903.03411.

Benjamin Estermann, Luca A. Lanzendörfer, Yannick Niedermayr, and Roger Wattenhofer. 2024. Puzzles: A benchmark for neural algorithmic reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 127059–127098. Curran Associates, Inc.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2023. Chessgpt: Bridging policy learning and language modeling. *Preprint*, arXiv:2306.09200.

Michael R Garey and David S Johnson. 2002. *Computers and intractability*, volume 29. wh freeman New York.

Jonas Golde, Patrick Haller, Fabio Barth, and Alan Akbik. 2025. Mastermindeval: A simple but scalable reasoning benchmark. *Preprint*, arXiv:2503.05891.

Google. 2025. Gemini 2.5: Our most intelligent ai model.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh

Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jiayi Gui, Yiming Liu, Jiale Cheng, Xiaotao Gu, Xiao Liu, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2024a. Logicgame: Benchmarking rule-based reasoning abilities of large language models. *Preprint*, arXiv:2408.15778.

Jiayi Gui, Yiming Liu, Jiale Cheng, Xiaotao Gu, Xiao Liu, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2024b. Logicgame: Benchmarking rule-based reasoning abilities of large language models. *Preprint*, arXiv:2408.15778.

Andrea Høfler. 2014. Smt solver comparison. *Graz, July*, 17.

Chenghao Huang, Yanbo Cao, Yinlong Wen, Tao Zhou, and Yanru Zhang. 2024. Pokergpt: An end-to-end lightweight solver for multi-player texas hold'em via large language model. *Preprint*, arXiv:2401.06781.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. Boardgameqa: A dataset for natural language reasoning with contradictory information. *Advances in Neural Information Processing Systems*, 36:39052–39074.

Richard E. Korf. 1990. Real-time heuristic search. *Artificial Intelligence*, 42(2):189–211.

Rhyd Lewis. 2007. Metaheuristics can solve sudoku puzzles. *Journal of Heuristics*, 13:387–401.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. Zebralogic: On the scaling limits of llms for logical reasoning. *Preprint*, arXiv:2502.01100.

Bill Yuchen Lin, Ziyi Wu, Yichi Yang, Dong-Ho Lee, and Xiang Ren. 2021. Riddlesense: Reasoning about riddle questions featuring linguistic creativity and commonsense knowledge. *Preprint*, arXiv:2101.00376.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2023. Agentbench: Evaluating llms as agents. *Preprint*, arXiv:2308.03688.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Microsoft, :, Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi ling Chen, Qi Dai, and 57 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *Preprint*, arXiv:2503.01743.

Chinmay Mittal, Krishna Kartik, Mausam, and Parag Singla. 2025. Fcorebench: Can large language models solve challenging first-order combinatorial reasoning problems? *Preprint*, arXiv:2402.02611.

Adithya Murali, Atharva Sehgal, Paul Krogmeier, and P. Madhusudan. 2022. Composing neural learning and symbolic reasoning with an application to visual discrimination. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, IJCAI-2022, page 3358–3365. International Joint Conferences on Artificial Intelligence Organization.

David Noever and Ryerson Burdick. 2021. Puzzle solving without search or human knowledge: An unnatural language approach. *Preprint*, arXiv:2109.02797.

OpenAI, :, Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, Jerry Tworek, Lorenz Kuhn, Lukasz Kaiser, Mark Chen, Max Schwarzer, Mostafa Rohaninejad, Nat McAleese, and 7 others. 2025. Competitive programming with large reasoning models. *Preprint*, arXiv:2502.06807.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024a. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2025a. Introducing gpt-4.1 in the api.

OpenAI. 2025b. Openai o3 and o4-mini system card.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Qwen. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

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

Yufan Ren, Konstantinos Tertikas, Shalini Maiti, Junlin Han, Tong Zhang, Sabine Süsstrunk, and Filippos Kokkinos. 2025. Vgrp-bench: Visual grid reasoning puzzle benchmark for large vision-language models. *Preprint*, arXiv:2503.23064.

Abdelrahman Sadallah, Daria Kotova, and Ekaterina Kochmar. 2025. Are llms good cryptic crossword solvers? *Preprint*, arXiv:2403.12094.

Tal Schuster, Ashwin Kalyan, Oleksandr Polozov, and Adam Tauman Kalai. 2021. Programming puzzles. *Preprint*, arXiv:2106.05784.

Dennis Shasha. 2017. Ruby risks. *Communications of the ACM*, 60(7):104–104.

Dennis Shasha. 2022a. Card nim. *Communications of the ACM*, 65(10):96–96.

Dennis Shasha. 2022b. Exclusivity probes. *Communications of the ACM*, 65(7):96–ff.

Dennis Shasha. 2022c. Maximal cocktails. *Communications of the ACM*, 66(1):112–112.

Dennis Shasha. 2023. Tidy towers. *Communications of the ACM*, 66(10):116–ff.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and 1 others. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

David K. Smith. 2007. Dynamic programming and board games: A survey. *European Journal of Operational Research*, 176(3):1299–1318.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Graham Todd, Tim Merino, Sam Earle, and Julian Togelius. 2024. Missed connections: Lateral thinking puzzles for large language models. *Preprint*, arXiv:2404.11730.

Nemika Tyagi, Mihir Parmar, Mohith Kulkarni, Aswin RRV, Nisarg Patel, Mutsumi Nakamura, Arindam Mitra, and Chitta Baral. 2024. Step-by-step reasoning to solve grid puzzles: Where do llms falter? *Preprint*, arXiv:2407.14790.

Clinton J. Wang, Dean Lee, Cristina Menghini, Johannes Mols, Jack Doughty, Adam Khoja, Jayson Lynch, Sean Hendryx, Summer Yue, and Dan Hendrycks. 2025. Enigmaeval: A benchmark of long multimodal reasoning challenges. *Preprint*, arXiv:2502.08859.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. 2024a. Smartplay: A benchmark for llms as intelligent agents. *Preprint*, arXiv:2310.01557.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. 2024b. Smartplay: A benchmark for llms as intelligent agents. *Preprint*, arXiv:2310.01557.

xAI. 2025. Grok 3 beta — the age of reasoning agents.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Zeyu Zhang, Zijian Chen, Zicheng Zhang, Yuze Sun, Yuan Tian, Ziheng Jia, Chunyi Li, Xiaohong Liu, Xiongkuo Min, and Guangtao Zhai. 2025. Puzzlebench: A fully dynamic evaluation framework for large multimodal models on puzzle solving. *Preprint*, arXiv:2504.10885.

Jingmiao Zhao and Carolyn Jane Anderson. 2023. Solving and generating npr sunday puzzles with large language models. *Preprint*, arXiv:2306.12255.

Contents

A PUZZLEPLEX

A.1 Dataset Overview

Name	Scenario	Reward	Data	Main Reasoning
SudoKill	Two-player	Deterministic	Text	Logical, Spatial
TidyTower (Shasha, 2023)	Single-player	Deterministic	Text	Spatial
CardNim (Shasha, 2022a)	Two-player	Deterministic	Text	Numerical, Logical
OptimalTouring	Single-player	Deterministic	Text	Numerical
CountMaximalCocktails (Shasha, 2022c)	Single-player	Deterministic	Text	Logical
MaxMaximalCocktails	Two-player	Deterministic	Text	Logical
ExclusivityParticles (Shasha, 2022b)	Two-player	Deterministic	Text	Numerical, Spatial
ExclusivityProbes	Single-player	Stochastic	Text	Numerical, Spatial
RubyRisks (Shasha, 2017)	Single-player	Stochastic	Text	Numerical, Logical
BeatOrBombSto.	Two-player	Stochastic	Text	Logical, Numerical
MaxTarget	Single-player	Stochastic	Text	Logical, Numerical
LargerTarget	Two-player	Stochastic	Text	Logical, Numerical
Superply	Two-player	Deterministic	Text	Numerical, Spatial
SudoKill M.	Two-player	Deterministic	Text-Image	Visual, Logical
Superply M.	Two-player	Deterministic	Text-Image	Visual, Numerical

Table 5: Overview of Puzzle Games.

A.2 Example of Simulator

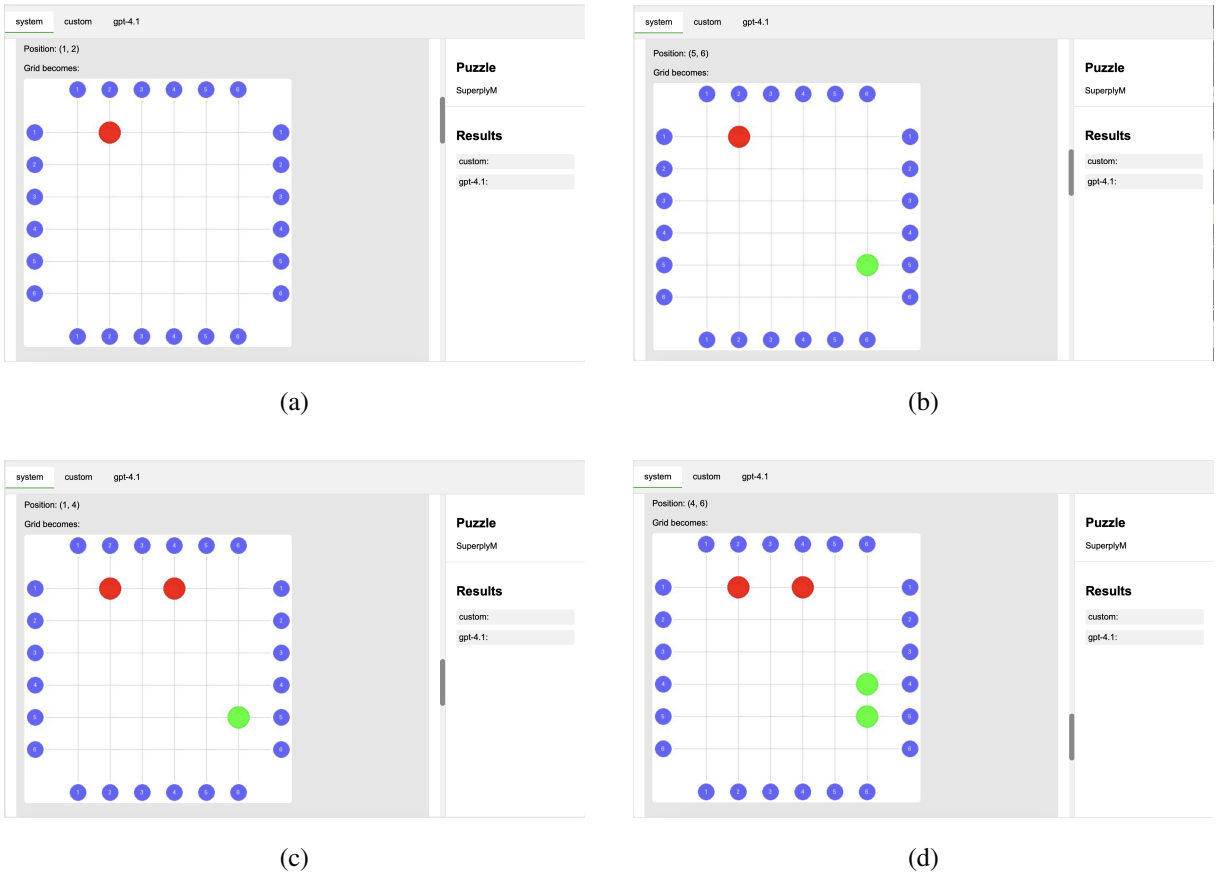

Figure 4: Overview of **Simulator**. The purpose of the Simulator is to present a history of the moves for a given puzzle for review by people. The representation of that history will differ for each kind of puzzle and the particular steps will depend on the methods used. SuperplyM is a two-player puzzle whose pedagogical goal for people is to teach arithmetic (e.g. multiplication). Play alternates between red and green players. When the red player responds correctly to a hint (panels a and c), the location chosen by that player turns red. When the green player responds (panels b and d), the location chosen by the green player turns green. In this image, we see a sequence of four moves, two by red and two by green, illustrating the history of moves taken by each method.

A.3 Breakdown Description of Puzzles

SudoKill

Rule
Sudokill is a competitive two-player variant of the classic Sudoku game. Like standard Sudoku, the game is played on a grid where the objective is to fill each row, column, and subgrid with the numbers from 1 to n, where n is the size of the row or column, without repeating any number in the same row, column, or subgrid.

In Sudokill, players take turns placing a number into an unoccupied cell. The first player can choose any empty cell to start the game. After that, each player must place their number in an unoccupied cell that lies in the **same row or column** as the last move made by their opponent. If there are no such cells available, the player may choose any unoccupied cell on the board.

A move is considered **invalid** if it violates standard Sudoku rules (i.e., placing a number that already appears in the same row, column, or subgrid), or if it is made in a cell not allowed by the row/column constraint described above. The first player to make an invalid move **loses** the game.

Example
If the current grid is

```
[6, 8, 4, 5, 1, 3, 2, 7, 9],
[5, 9, 7, 6, 2, 0, 1, 8, 0],
[2, 3, 1, 4, 8, 7, 6, 5, 0],
[9, 1, 2, 7, 6, 4, 8, 0, 3],
[4, 6, 8, 3, 0, 1, 7, 2, 5],
[7, 5, 3, 2, 9, 8, 4, 1, 6],
[8, 4, 5, 1, 3, 2, 9, 6, 7],
[1, 0, 6, 9, 0, 5, 0, 3, 8],
[3, 2, 0, 0, 7, 0, 5, 4, 0]
```

and now is your turn and the previous move by the opponent is to fill the cell at (0, 8) with the value 9. So now the cells you can place a number are [(1,8), (2,8), (8,8)] because you can only place a number in the same row or column as the last move.

If the current grid is

```
[6, 8, 4, 5, 1, 3, 2, 7, 9],
[5, 9, 7, 6, 2, 0, 1, 8, 0],
[2, 3, 1, 4, 8, 7, 6, 5, 0],
[9, 1, 2, 7, 6, 4, 8, 0, 3],
[4, 6, 8, 3, 0, 1, 7, 2, 5],
[7, 5, 3, 2, 9, 8, 4, 1, 6],
[8, 4, 5, 1, 3, 2, 9, 6, 7],
[1, 0, 6, 9, 0, 5, 0, 3, 8],
[3, 2, 0, 0, 7, 0, 5, 4, 1]
```

and now is your turn and the previous move by the opponent is to fill the cell at (0, 8) with the value 9. Now you can fill the cell (1, 8) with the value 4 to win this game because after you fill the cell (1, 8) with the value 4, the opponent can only fill the cell (2, 8) and (1, 5), but no matter which value the opponent fills in these two cells will violate the rules.

Figure 5: Description of SudoKill.

TidyTower

Rule
Tidy Tower is a single-player puzzle involving a vertical stack of cubes, each with four colored sides arranged in a fixed clockwise order: **Red** (R), **Yellow** (Y), **Blue** (B), and **Green** (G). The player's objective is to transform the tower such that all cubes display the same color on their front face—this state is referred to as a tidy tower.

Two types of operations are allowed to manipulate the tower:

Rotate: When a player rotates a cube at a certain index, that cube and all cubes above it rotate together in a clockwise direction. A single rotation shifts the front-facing side of a cube to the next color in the clockwise sequence. For example, rotating once changes a cube with front face "R" to "Y", and so on. Rotating four times returns it to the original orientation.

Rotate with Holding: A player can also rotate a cube while holding a cube above it. This operation rotates only the selected cube and all cubes below it, while the held cube and any cubes above it remain in place.

Example
The initial setting is: RGBYRGBYBGBGBG. Can you make this tower tidy in eight moves or less? Solution for eight moves:
RGBYRGBYBGBGBG → (rotate by one position at position 1 and not hold at position 2)
RRGBYRGBGRGRGR → (rotate by one position at position 2 and hold at position 3)
RRRBYRGBGRGRGR → (rotate by two positions at position 3 and hold at position 4)
RRRRYRGBGRGRGR → (rotate by one at position 4 and hold at position 5)
RRRRRRGBGRGRGR → (rotate by one at position 6 and hold at position 9)
RRRRRRRGRRGRGR → (rotate by one at position 7 and hold at position 8)
RRRRRRRRRRGRGR → (rotate by one at position 10 and hold at position 11)
RRRRRRRRRRRRGR → (rotate by one at position 12 and hold at position 13)
RRRRRRRRRRRRR → Done

Figure 6: Description of TidyTower.

CardNim

Rule
Card Nim is a two-player turn-based game played with a shared pile of stones and individual hands of number cards. At the start of the game, both players receive a set of cards, each card displaying a positive integer. A single pile of stones is placed at the center of the board.

On each turn, a player must play one of their cards to remove exactly that number of stones from the pile. A card can only be played if its value is **less than or equal** to the number of stones remaining. Once a card is used, it cannot be reused. The two players take turns alternately.

The objective is to be the player who removes the **last stone** from the pile. However, if a player is unable to play any card on their turn—because all of their remaining cards are greater than the number of remaining stones—they **lose the game** immediately.

Example
For example, suppose there are five stones left and each of the two players you and your opponent has three cards with 1, 2,and 3, respectively. You goes first. Who wins?
Your opponent wins. If you removes 2 or 3, then opponent can win immediately with 3 or 2 respectively.
So, you removes 1. Now your opponent removes 3, leaving 1. Now you has only cards with numbers greater than 1 so you lose.

Figure 7: Description of CardNim.

OptimalTouring

Rule
Optimal Touring is a route optimization puzzle in which a player must plan a one-day tour across a set of tourist sites. Each site is defined by the following attributes:
- A **location** represented by street and avenue coordinates.
- A **fixed visiting time** (in minutes) that must be spent at the site.
- A **value**, indicating the reward or importance of visiting the site.
- A **visiting window** specified by a start and end hour (in 24-hour format), representing when the site is accessible.

The player's objective is to **maximize the total value** of visited sites while adhering to time and location constraints.

The total time spent on the tour includes:
- The **visit time** required at each site.
- The **travel time** between consecutive sites, computed using **Manhattan distance** (i.e., the sum of the absolute differences in street and avenue numbers).

The tour can start at **any site**, but each site must be visited **within its allowed time window**, and the cumulative time (including both visiting and travel) must respect this schedule. Once a site is visited, its value is counted toward the total.

Example
Here is an example of data:

Site	Avenue	Street	Desired Time	Value	Begin Hour	End Hour
1	50	96	114	3	6	12
2	8	23	190	186	9	17
3	88	69	218	3	9	12
4	0	95	101	86	6	12
5	1	48	192	199	5	12

If you start visit cite 5 at 5:00, then go to site 2, then the hour is after 12:00, and the value you get is 199 + 186 = 385.

Figure 8: Description of OptimalTouring.

CountMaximalCocktails

Rule
Count Maximal Cocktails is a combinatorial puzzle inspired by drug treatment for orphan diseases, where the objective is to discover safe and effective drug combinations. Each drug is represented as a **node** in an undirected graph. Pairs of drugs that should not be combined due to harmful interactions are represented as **edges** connecting the respective nodes.

A **cocktail** is a subset of drugs that can be administered together safely—meaning no two drugs in the subset have a harmful interaction. In graph theory, such a subset corresponds to an **independent set**, where no two nodes are directly connected by an edge.

The goal is to identify **all maximal independent sets** of the graph, referred to in this context as **maximal cocktails**. A set is **maximal** if it is an independent set and no additional drug can be added to it without introducing a harmful interaction (i.e., violating independence). Note that "maximal" does not mean "maximum in size"; rather, it means that the set cannot be extended further while maintaining its validity.

Two difficulty settings are defined:
In the **easy** level, only the number of maximal cocktails needs to be determined.
In the **normal** level, the task is to list all maximal cocktails explicitly.

The input includes a list of **drugs (nodes_list)** and a **list of harmful interactions (edges_list)**, where each edge is a pair of drug identifiers indicating a conflict.

Example
Suppose the drug list is [1, 2, 3, 4] and the bad interaction list is [(1, 2)].

The maximal cocktails are [1, 3, 4] and [2, 3, 4], and the number of maximal cocktails is 2.

Figure 9: Description of CountMaximalCocktails.

MaxMaximalCocktails

Rule
Max Maximal Cocktails is a strategic two-player game played on a graph where nodes represent **drugs**, and edges represent **harmful interactions** between drug pairs. The game's core objective is to manipulate the structure of the graph by **adding edges**, while maintaining or increasing the number of valid drug combinations—called **maximal cocktails**.

A maximal cocktail is defined as a **maximal independent set** in the graph: a set of drugs in which no two drugs have a harmful interaction, and to which no more drugs can be added without creating a conflict.

At the beginning of the game, a list of nodes (drugs) is provided with **no edges**, meaning all combinations are potentially valid. Players take turns, and on each turn, a player **adds an edge** between two distinct nodes. The added edge represents the discovery or introduction of a harmful interaction between the two corresponding drugs.

The key rule is that **a move is only legal if it does not decrease** the current number of maximal cocktails. The first player who adds an edge that causes a decrease in the number of maximal cocktails loses the game.

Example
Suppose the node list is [1, 2, 3], and you are the first player, you can add the edge (1, 2), then the number of maximal cocktails is 2, which is larger than the number of maximal cocktails without the edge (1, 2), which is 1. So this addition is legal.

But if your opponent adds the edge (2, 3) after you add the edge (1, 2), then the number of maximal cocktails is 3, which is also legal.

After that, you will lose since you cannot add any edge to increase the number of maximal cocktails.

Figure 10: Description of MaxMaximalCocktails.

ExclusivityParticles

Rule
Exclusivity Particles is a two-player combinatorial game played in a **d-dimensional binary space**, often conceptualized as the vertices of a **d-dimensional hypercube**. Each coordinate in this space is binary—either 0 or 1—representing discrete states along each dimension (e.g., spin up or down).

Players take turns placing **particles** at positions in this space. Each particle occupies a unique vertex of the hypercube. A strict exclusion principle governs the game: Any two particles must differ in at least **k dimensions**, meaning their **Hamming distance** must be **greater than or equal to k**. The Hamming distance is calculated as the number of differing coordinates between two binary vectors.

The game proceeds as follows:
- The **first player** places a particle at any position in the d-dimensional binary space.
- The **second player** then places another particle at a different position that satisfies the minimum distance condition with respect to all previously placed particles.
- Players alternate turns.
- A player **loses** if they cannot place a new particle that maintains the required distance **from all previously placed particles**.

Example
If the dimension is 3 and the required distance is 2, and you are the first player, you could place the first particle at [0, 0, 0].

The second player could then place the second particle at [0, 1, 1]. If you place the third particle at [1, 0, 1], the second player cannot place a fourth particle that satisfies the condition and would lose.

Figure 11: Description of ExclusivityParticles.

ExclusivityProbes

Rule
Exclusivity Probes is a deductive search game played in a **d-dimensional binary space**, conceptually represented as a **d-dimensional hypercube**, where each position is a binary vector of length d (each dimension having value 0 or 1).

There are exactly num_particles particles hidden in this space, and they obey a strict **exclusion principle**: any two particles must differ in at least k dimensions, meaning their **Hamming distance** must be greater than or equal to k. The Hamming distance between two positions is the number of coordinates in which they differ.

The player interacts with the environment by making probes. A probe is a query at a specific position in the hypercube. The response will be:
- "yes" if there is a particle at that exact position,
- "no" otherwise.

The objective is to identify the exact locations of all particles using **as few probes as possible**.

Example
If the dimension is 2, the number of particles is 2, and the distance is 1.

You can probe the position [0, 0], and if the response is 'yes', we only need one more probe to find the other particle because the particles can be either at locations [0, 0] and [1, 1] or at [0, 1] and [1, 0].

If the response is 'no', we may need 3 more probes to find all the particles.

Figure 12: Description of ExclusivityProbes.

RubyRisks

Rule
Ruby Risks is a sequential deduction game involving a set of num_boxes hidden containers, each holding an **unknown number of identical rubies**. The total number of rubies across all boxes is known in advance and given as total_rubies.

Each turn, the player submits a **single request**: a number of rubies they wish to take from **the next unopened box**. Boxes are opened in fixed left-to-right order, one per turn.

The outcome of a request depends on the number of rubies hidden in the box:
- If the request is **less than or equal** to the number of rubies in the box, the player **successfully collects** that amount.
- If the request is **greater than** the number of rubies in the box, the request fails, and the player receives **nothing** from that box.

The game proceeds turn by turn, with each turn corresponding to a new box. The goal is to **maximize the total number of rubies collected** across all turns.

Example
Suppose there are 3 boxes, and the hidden rubies in each box are: [11, 9, 10]. Total rubies = 30.

Turn 1: You request 10 rubies.
Feedback: 10 (successfully take 10 rubies).

Turn 2: You request 8 rubies.
Feedback: 8 (successfully take 8 rubies).

Turn 3: You request 12 rubies.
Feedback: 0 (because 12 > 10, so you get nothing from that box).

Total rubies collected so far: 18.

Figure 13: Description of RubyRisks.

BeatOrBombSto

Rule
Beat Or Bomb Sto. is a two-player card game where players tactically choose when to **compete** or **give up** with the cards in their hands to maximize their total score across several rounds.

At the beginning, each player is given a set of num_cards, which may differ in composition but are balanced so that the **total value** of each set is the same. Card values are assigned as follows:
- Numeric cards (2–10) are worth their face value.
- Jacks, Queens, Kings, and Aces have values of 11, 12, 13, and 1, respectively.

Each round proceeds as follows:
- Both players simultaneously select one card from their remaining set and decide whether to compete with it or give it up.
- This decision is **private**—neither player knows the other's card or choice until both have confirmed.
- Regardless of the choice, the selected card is **removed** from the player's hand.

Scoring rules:
- If both players compete, the player with the **higher** card earns points equal to **both card values combined**. The other player earns nothing.
- If both players give up, **no points are awarded**.
- If one player competes while the other gives up, the competing player earns points **equal to their card's value**. The giving-up player earns nothing.

The game continues until all cards are used. The player with the **most points** at the end wins.

Example
In one of the rounds, if you choose to play the card '5' and compete, your opponent plays the card 'K' and give up, you will get 5 points and your opponent gets nothing.

Figure 14: Description of BeatOrBombSto.

MaxTarget

Rule
Max Targetis a probabilistic decision-making game where the player must choose bags of coins over a fixed number of turns to maximize the **total value of collected coins**. The game consists of bag_count bags, each containing coins with specific known values (e.g., [1, 2], [3, 4]), but **the order of the bags is randomized** before gameplay begins.

At the start, the player is informed of:
- The list of coin values contained in each bag.
- The **total number of picks** allowed during the game (max_guess).

Each turn proceeds as follows:
- The player chooses a **bag index**.
- One **random coin** is drawn from the chosen bag and added to the player's total score.
- The drawn coin is then **removed** from that bag.
- Over time, based on the drawn coins, the player can infer which observed bag maps to which known configuration.

Example
If you're told the bags contain [1, 2] and [3, 4], and the total number of picks is 2.

If you pick bag 0 and get a coin value of 4, then in the next turn, you will know that bag 0 contains [3, 4] and bag 1 contains [1, 2], and value 4 in bag 0 is removed and remaining values are [3].

So, if you pick bag 0 again, you will get a coin value of 3, which is bigger than the coin value of bag 1. So, you should pick bag 0 again to maximize your score.

Figure 15: Description of MaxTarget.

LargerTarget

Rule
Larger Target is a two-player competitive coin-picking game. There are bag_count bags, each containing a known list of coin values, but the order of the bags is **randomized** at the start of the game. Players alternate turns, each making a total of max_guess picks across the game.

Each turn proceeds as follows:
- The player selects a **bag index**.
- A **random coin** is drawn from the chosen bag and removed from it.
- The value of the drawn coin is added to the player's total score.

The objective is to **accumulate a higher total coin value** than the opponent by making informed decisions about which bag to choose. Since the order of bags is shuffled, players must deduce which real bag corresponds to each index by observing the coins drawn—both by themselves and their opponent.

Example
If you're told the bags contain [1, 2] and [3, 4], and the total number of picks is 2.

If your opponent pick bag 0 and get a coin value of 3, then in your turn, you will know that bag 0 contains [3, 4] and bag 1 contains [1, 2], and value 3 in bag 0 is removed and remaining values are [4]. So, if you pick bag 0 again, you will get a coin value of 4, which is bigger than the coin value of bag 1. So, you should pick bag 0 to make your score higher than your opponent.

Figure 16: Description of LargerTarget.

Superply

Rule
Superply is a two-player competitive path-building game played on a 1-indexed grid-based board. The grid is initially filled with zeros and players take turns selecting valid positions based on system-provided **mathematical hints**.

Each player has a unique path-building objective:
- Player 1 aims to build a continuous path of their claimed squares (marked with value 1) from the **left** edge of the grid to the **right**.
- Player 2 aims to build a path (marked with value 2) from the **top** edge to the **bottom**.

A valid path is a sequence of adjacent same-value cells, where adjacency includes both **sidewise and diagonal** (corner) neighbors.

Each turn proceeds as follows:
- The system provides a **hint**, such as a condition on the sum or product of the row and column indices (e.g., "sum < 10", "product contains digit 6").
- The player selects a grid cell (row, column) that: 1) Is currently **unoccupied** (i.e., value is 0); 2) **Satisfies the hint condition**.
- If the selection is **valid**, the cell is **updated** to reflect the player's value (1 or 2). Otherwise, the move is **skipped and the turn passes** to the opponent.

The game ends when a player successfully builds a full path satisfying their objective. The first to do so wins.

Example
If the hint is "product contains digit 6," and the grid is as follows:

```
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0]
```

If you are Player 1, you can select the position (1, 6), (6, 1), (2, 3), (3, 2) or (6, 6) because the product of the row and column indices is 6, 6, 6, 6 and 36, respectively, and they all contain the digit 6.

If you choose the position (6, 6), the grid becomes:

```
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1]
```

Figure 17: Description of Superply.

B Experimental Setup

B.1 LLMs Configuration

Model	Creator	Version	Access Time	License	Input Modalities
Deepseek-R1	Deepseek	Deepseek-R1	2025.1	Open-source	text
o4-mini	OpenAI	o4-mini-medium	2025.4	Proprietary	text & image
Gemini-2.5-pro	Google	gemini-2.5-pro-preview-03-25	2025.3	Proprietary	text & image
QwQ-32B	Alibaba	QwQ-32B	2025.3	Open-source	text
grok-3-mini	xAI	grok-3-mini	2025.2	Proprietary	text
Deepseek-V3	Deepseek	Deepseek-V3	2024.12	Open-source	text
GPT-4.1	OpenAI	gpt-4.1-2025-04-14	2025.4	Proprietary	text & image
Qwen2.5-VL-72B	Alibaba	Qwen2.5-VL-72B-Instruct	2025.2	Open-source	text & image
Llama-3.3-70B	Meta	Llama-3.3-70B-Instruct	2024.12	Open-source	text
Gemma-3-27B	Google	gemma-3-27b-it	2025.3	Open-source	text & image
Phi-4-multimodal	Microsoft	Phi-4-multimodal-instruct	2025.3	Open-source	text & image

Table 6: Details of the LLMs evaluated in PUZZLEPLEX.

B.2 Costomized Model Configuration

Puzzle Name	Baseline Strategy	
	Easy	**Normal**
SudoKill	Random	Greedy
TidyTower	Dynamic Programming	Dynamic Programming
CardNim	Random	Dynamic Programming
OptimalTouring	Simulated Annealing Algorithm	Simulated Annealing Algorithm
CountMaximalCocktails	Brute-force	Brute-force
MaxMaximalCocktails	Random	Brute-force
ExclusivityParticles	Brute-force	Greedy
ExclusivityProbes	Random	Greedy
RubyRisks	Monte-Carlo Tree Search	Monte-Carlo Tree Search
BeatOrBombSto.	Random	Greedy
MaxTarget	Greedy	Greedy
LargerTarget	Random	Greedy
Superply	Random	Searching

Table 7: Overview of puzzle games and their basic strategies. For text-image puzzles, we apply strategies similar to those used in corresponding text-only puzzles.

B.3 Implementation Details of Model Inference

We use APIs to evaluate several models: Deepseek-R1, o4-mini, Gemini-2.5-pro, grok-3-mini, Deepseek-V3, GPT-4.1, and Phi-4-multimodal. For other models, we utilize Hugging Face Transformers (Wolf et al., 2020) inference on 8 $\times$ H100 and 8 $\times$ A100.

B.4 Implementation Details of Elo Score

Each player begins with an initial rating $R = 1000$. After a match between player A and player B, player A's updated rating is given by $R'_A = R_A + K \cdot (S_A - E_A)$ where R_A and R_B are the current Elo ratings of players A and B, respectively. $K = 32$ is the update constant we set. $S_A \in \{1, 0.5, 0\}$ is the actual result of the game (1 = win, 0.5 = draw, 0 = loss). E_A is the expected score for player A, computed as $E_A = \frac{1}{1+10^{(R_B - R_A)/400}}$.

B.5 Operation Extraction

In an instruction-based as well as a code-based setting, the raw output of the LLM may not be in a correct format. For each turn of an LLM, PUZZLEPLEX allows the LLM up to five attemps to generate a move with the correct format (based on a regular expression checker). As soon as an attempt generates a correct format, the result is sent to the state transition engine of PUZZLEPLEX. If none of attempts generates a move having the correct format, that LLM loses the game.

B.6 The Cost of Experiments

Model	Instr.-based Cost ($)	Code-based Cost ($)	Total Cost ($)	Instr.-based GPU Hrs	Code-based GPU Hrs	Total GPU Hrs
GPT-4.1	162.83	7.59	170.42	–	–	–
o4-mini	90.03	3.27	93.30	–	–	–
Gemini-2.5-pro	556.86	33.03	589.89	–	–	–
grok-3-mini	20.16	5.21	25.37	–	–	–
Deepseek-R1	37.41	14.78	52.19	–	–	–
Deepseek-V3	27.99	1.22	29.21	–	–	–
Phi-4-multimodal	8.82	0.36	9.18	–	–	–
QwQ-32B	–	–	–	50.82	6.76	57.58
Llama-3.3-70B	–	–	–	48.17	2.02	50.19
Gemma-3-27B	–	–	–	10.17	3.89	14.06
Qwen2.5-VL-72B	–	–	–	56.30	1.96	58.26
Total	903.10	65.46	968.56	165.46	14.63	180.09

Table 8: Experiment costs estimated across both instruction-based and code-based settings using two approaches: API-based models are priced in USD, and GPU-based models are quantified in NVIDIA H100 GPU hours.

B.7 Code-based Prompt Template

Code Template Prompt

You are about to play a game called {puzzle name} against an opponent.

{puzzle rules}

{code input description and format}

{code output description and format}

{code template}

{required LLM output format}

Figure 18: The code template prompt in code-based setting.

C Experiment Results

C.1 Elo Score Results

Model	Single-Player Det.		Two-Player Det.		Score
	Easy	Normal	Easy	Normal	
Custom	1250.9 ±504.5	1229.9 ±529.1	1060.8 ±309.8	1081.3 ±347.2	1134.6 ±126.4
Deepseek-R1	1146.6 ±445.8	1084.3 ±379.5	1163.9 ±122.0	1185.1 ±126.2	1152.4 ±63.2
o4-mini	1094.9 ±549.8	1120.0 ±649.5	1165.7 ±96.0	1156.4 ±100.3	1140.9 ±74.9
Gemini-2.5-pro	1082.1 ±494.9	1046.3 ±241.2	1145.0 ±64.3	1117.8 ±136.7	1106.2 ±58.1
QwQ-32B	1074.1 ±268.5	988.2 ±269.8	1160.0 ±111.5	1122.9 ±84.4	1100.1 ±54.7
grok-3-mini	882.4 ±366.0	911.1 ±254.8	1145.9 ±190.1	1120.6 ±225.1	1044.6 ±97.5
Deepseek-V3	980.2 ±60.4	950.7 ±89.1	989.6 ±163.4	967.7 ±87.9	973.7 ±42.7
GPT-4.1	1041.8 ±321.8	1042.5 ±321.4	1024.8 ±137.2	1044.7 ±111.1	1037.5 ±53.1
Qwen-2.5-VL-72B	905.2 ±349.7	950.7 ±282.0	762.6 ±117.0	817.2 ±194.8	841.7 ±73.0
Llama-3.3-70B	863.3 ±381.6	891.9 ±266.0	826.9 ±108.3	792.1 ±99.6	835.0 ±52.9
Gemma-3-27B	858.8 ±397.7	918.3 ±217.5	797.5 ±114.3	797.7 ±103.2	831.7 ±55.8
Phi-4-multimodal	819.8 ±294.0	866.0 ±322.9	757.2 ±144.9	796.5 ±97.6	801.6 ±55.8

Table 9: Comparison of Elo scores (mean ± 95% CI) across various models on single-player and two-player deterministic puzzles, categorized by difficulty.

Model	TidyTower		OptimalTouring		CountMaximalCocktails	
	Easy	Normal	Easy	Normal	Easy	Normal
Custom	1481.3	1464.0	1098.1	1047.6	1173.2	1178.2
o4-mini	956.1	952.8	1350.1	1421.3	978.4	985.8
Deepseek-r1	956.2	952.4	1312.6	1251.6	1170.9	1048.8
Deepseek-V3	956.2	952.2	1004.8	914.1	979.7	985.8
GPT-4.1	956.1	953.0	1190.8	1190.9	978.4	983.8
grok-3-mini	956.2	952.5	712.7	794.3	978.2	986.5
Gemini-2.5-pro	956.1	952.7	1311.8	1146.5	978.3	1039.8
QwQ-32B	956.3	952.0	1168.7	1110.3	1097.2	902.4
Qwen-2.5-VL-72B	956.4	1012.7	746.0	819.8	1013.2	1019.8
Llama-3.3-70B	956.3	951.9	686.0	768.3	947.5	955.5
Gemma-3-27B	956.4	951.8	674.1	819.0	946.1	984.2
Phi-4-multimodal	956.3	952.1	744.4	716.5	758.9	929.4

Table 10: Performance comparison in Elo ratings of large language models on three single-player deterministic puzzles, categorized by difficulty.

Model	CardNim		SudoKill		MaxMaximalCocktails		ExclusivityParticles		Superply	
	Easy	Normal	Easy	Normal	Easy	Normal	Easy	Normal	Easy	Normal
Custom	688.9	697.6	1155.1	1355.5	1274.6	1106.5	1257.2	1331.1	928.2	915.8
o4-mini	1206.8	1153.9	1149.5	1048.4	1039.1	1124.0	1196.9	1188.1	1236.2	1267.8
Deepseek-r1	1245.8	1265.2	1050.3	1266.1	1110.9	1025.3	1126.8	1146.8	1285.6	1222.4
Deepseek-V3	1042.0	1002.4	1198.3	964.0	898.5	1012.8	913.8	846.3	895.4	1013.1
GPT-4.1	984.1	1084.2	1071.9	1033.7	999.3	990.0	885.7	941.5	1182.9	1173.9
grok-3-mini	1340.8	1313.8	1216.0	1260.2	935.0	864.6	1069.1	1026.6	1168.9	1137.7
Gemini-2.5-pro	1201.7	1221.1	1160.3	1090.6	1063.4	1109.0	1166.5	1216.1	1133.0	952.4
QwQ-32B	1252.2	1190.7	1132.3	1116.5	1021.2	1016.3	1175.4	1118.5	1219.1	1172.4
Qwen-2.5-VL-72B	679.4	789.9	705.5	659.5	873.3	1081.2	857.0	782.6	698.0	772.9
Llama-3.3-70B	839.8	847.7	789.3	653.9	971.4	847.8	786.8	811.6	747.1	799.3
Gemma-3-27B	890.8	752.7	662.3	771.3	875.7	936.1	771.4	806.1	787.4	722.3
Phi-4-multimodal	627.6	680.9	709.1	780.3	937.6	886.4	793.2	784.8	718.3	850.1

Table 11: Performance comparison in Elo ratings of large language models on five two-player deterministic puzzles. Scores are reported in Elo ratings for both Easy and Normal difficulty settings.

C.2 Results of Different Prompting Strategies

Model	TidyTower		SudoKill	
	Easy	Normal	Easy	Normal
GPT-4.1	0.00	0.00	0.30	0.10
w/ 1-shot	0.00	0.00	0.10	0.10
w/ ToT	0.60	0.80	0.30	0.10
w/o history	1.00	0.90	0.20	0.00
w/ legal candidates	0.60	0.00	0.60	0.70
o4-mini	0.00	0.00	0.40	0.20
w/ 1-shot	0.00	0.00	0.50	0.20
w/ ToT	0.80	0.70	0.40	0.40
w/o history	1.00	1.00	0.40	0.30
w/ legal candidates	1.00	0.60	0.50	0.40

Table 12: Performance of GPT-4.1 and o4-mini on TIDYTOWER and SUDOKILL puzzles under different prompting strategies.

C.3 Results of Multimodal Puzzles

Model	SudoKillM						SuperplyM					
	Easy			Normal			Easy			Normal		
	Score	vs. Text	vs. Custom	Score	vs. Text	vs. Custom	Score	vs. Text	vs. Custom	Score	vs. Text	vs. Custom
Gemini-2.5-pro	0.48	0.21	0.46	0.60	0.56	1.00	0.60	0.54	0.20	0.56	0.57	0.32
Gemma-3-27B	0.25	0.37	0.00	0.25	0.24	0.00	0.25	0.23	0.05	0.25	0.17	0.25
GPT-4.1	0.48	0.45	0.30	0.36	0.40	0.10	0.58	0.53	0.30	0.58	0.56	0.20
o4-mini	0.72	0.62	0.20	0.54	0.59	0.40	0.94	0.83	0.98	0.94	0.83	0.90
Phi-4-multimodal	0.20	0.10	0.00	0.35	0.14	0.00	0.25	0.09	1.00	0.27	0.24	0.30
Qwen-2.5-VL-72B	0.30	0.18	0.10	0.24	0.15	0.00	0.24	0.12	0.30	0.32	0.16	0.30

Table 13: Normalized score on SUDOKILLM and SUPERPLYM puzzles. Colors indicate the percentage change of the text-only or custom-strategy baseline versus the multimodal model score.

C.4 Breakdown Instruction-based Results of Puzzles

Model	TidyTower		OptimalTouring		CountMaximalCocktails	
	Easy	Normal	Easy	Normal	Easy	Normal
Custom	1.00	1.00	0.67	0.49	1.00	1.00
Deepseek-R1	0.00	0.00	0.91	0.75	1.00	0.70
o4-mini	0.00	0.00	0.93	0.92	0.40	0.40
Gemini-2.5-pro	0.00	0.00	0.91	0.82	0.40	0.50
QwQ-32B	0.00	0.00	0.81	0.46	0.80	0.30
grok-3-mini	0.00	0.00	0.10	0.27	0.40	0.40
Deepseek-V3	0.00	0.00	0.62	0.42	0.40	0.30
GPT-4.1	0.00	0.00	0.79	0.75	0.40	0.30
Qwen-2.5-VL-72B	0.00	0.20	0.21	0.11	0.50	0.40
Llama-3.3-70B	0.00	0.00	0.14	0.07	0.30	0.30
Gemma-3-27B	0.00	0.00	0.09	0.06	0.30	0.30
Phi-4-multimodal	0.00	0.00	0.14	0.00	0.00	0.10

Table 14: Instruction-based normalized scores of models on single-player deterministic puzzles, separated by difficulty.

Model	CardNim		SudoKill		MaxMaximalCocktails		ExclusivityParticles		Superply	
	Easy	Normal	Easy	Normal	Easy	Normal	Easy	Normal	Easy	Normal
Custom	0.16	0.25	0.75	0.86	0.83	0.64	0.80	0.86	0.43	0.40
Deepseek-R1	0.77	0.76	0.64	0.74	0.57	0.54	0.56	0.54	0.74	0.73
o4-mini	0.73	0.63	0.62	0.59	0.51	0.65	0.68	0.70	0.83	0.83
Gemini-2.5-pro	0.79	0.79	0.75	0.64	0.50	0.60	0.70	0.75	0.66	0.55
QwQ-32B	0.76	0.73	0.65	0.61	0.59	0.57	0.69	0.70	0.78	0.78
grok-3-mini	0.83	0.83	0.70	0.79	0.41	0.42	0.65	0.60	0.76	0.72
Deepseek-V3	0.53	0.54	0.69	0.57	0.45	0.52	0.46	0.30	0.45	0.56
GPT-4.1	0.47	0.46	0.45	0.40	0.47	0.44	0.25	0.38	0.53	0.56
Qwen-2.5-VL-72B	0.13	0.26	0.18	0.15	0.44	0.61	0.34	0.23	0.12	0.16
Llama-3.3-70B	0.34	0.39	0.23	0.17	0.46	0.40	0.31	0.33	0.22	0.26
Gemma-3-27B	0.27	0.19	0.13	0.24	0.36	0.38	0.15	0.25	0.23	0.17
Phi-4-multimodal	0.16	0.13	0.10	0.14	0.33	0.18	0.18	0.18	0.09	0.24

Table 15: Instruction-based normalized scores of models on two-player deterministic puzzles, separated by difficulty.

C.5 Instruction-based vs. Code-based

Model	Easy	Normal
Deepseek-R1	0.56 (-0.25)	0.50 (-0.32)
o4-mini	0.62 (-0.21)	0.54 (-0.19)
Gemini-2.5-pro	0.64 (-0.24)	0.58 (-0.23)
QwQ-32B	0.48 (-0.26)	0.33 (-0.25)
grok-3-mini	0.64 (-0.30)	0.56 (-0.28)
Deepseek-V3	0.40 (-0.27)	0.32 (-0.26)
GPT-4.1	0.46 (-0.25)	0.36 (-0.27)
Qwen-2.5-VL-72B	0.30 (-0.21)	0.24 (-0.19)
Llama-3.3-70B	0.32 (-0.16)	0.07 (-0.07)
Gemma-3-27B	0.06 (-0.05)	0.00 (-0.00)
Phi-4-multimodal	0.18 (-0.18)	0.20 (-0.20)

Table 16: Win rates of foundation models against the custom strategy in the instruction-based setting on two-player deterministic puzzles. The blue values in parentheses indicate the win rate difference (instruction-based minus code-based), highlighting performance drops in the code-based setting.

C.6 Play Statistics

Name	Total Play			Legal Play			Legal Play Percentage
	#Turns	#Tokens (R)	#Tokens (NR)	#Turns	#Tokens (R)	#Tokens (NR)	
SudoKill	3.26 ± 0.11	6281.64 ± 73.58	916.10 ± 24.00	7.40 ± 0.39	5362.98 ± 118.59	812.60 ± 38.14	0.12
TidyTower	1.00 ± 0.00	7636.68 ± 334.95	1273.59 ± 76.42	1.00 ± 0.00	7636.68 ± 334.95	1282.83 ± 77.25	0.99
CardNim	1.79 ± 0.02	5690.17 ± 143.70	492.85 ± 16.51	1.85 ± 0.02	5383.76 ± 156.37	444.68 ± 12.80	0.77
OptimalTouring	1.00 ± 0.00	17370.60 ± 615.55	1448.76 ± 70.53	1.00 ± 0.00	16375.78 ± 706.73	1563.17 ± 103.28	0.62
CountMaximalCocktails	1.00 ± 0.00	5217.48 ± 441.32	1228.92 ± 94.87	1.00 ± 0.00	5268.49 ± 459.68	1256.98 ± 96.57	0.96
MaxMaximalCocktails	1.56 ± 0.02	7286.40 ± 150.85	717.81 ± 21.12	1.51 ± 0.02	7870.05 ± 167.68	736.76 ± 23.98	0.78
ExclusivityParticles	7.09 ± 0.49	2823.25 ± 39.29	445.75 ± 6.75	–	–	–	0.00
Superply	10.13 ± 0.20	2519.22 ± 26.78	344.56 ± 4.17	11.39 ± 0.20	2514.00 ± 27.10	333.38 ± 3.40	0.84

Table 17: Statistics of play and legal play across puzzles in the instruction-based setting. (R) denotes reasoning models, and (NR) denotes non-reasoning models. A legal play refers to a game trajectory that ends with a legal termination status; for a detailed definition of termination status, please refer to § 4.4. #Turns indicates the number of turns per player. In two-player puzzles, the total number of rounds is the sum of turns across both players.

C.7 Win Probability Matrix

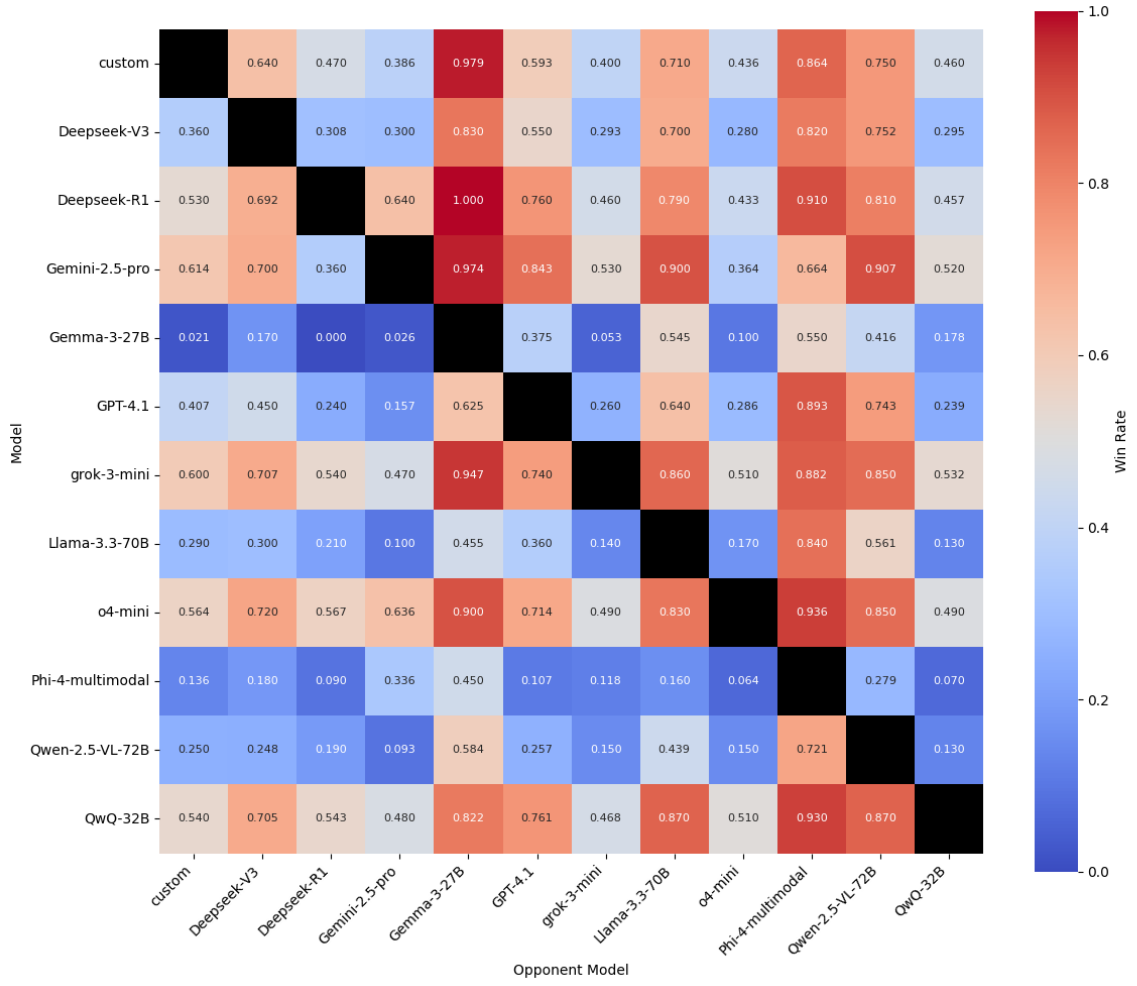

Figure 19: Win rate heatmap for 12 methods in two-player puzzles under the **instruction-based** setting. Each number represents the win rate of the row entry over the column entry, normalized to ignore ties. For example, for game instances that don't end in a tie, GPT-4.1 beats o4-mini 28.6% of the time and o4-mini beats GPT-4.1 71.4% of the time.

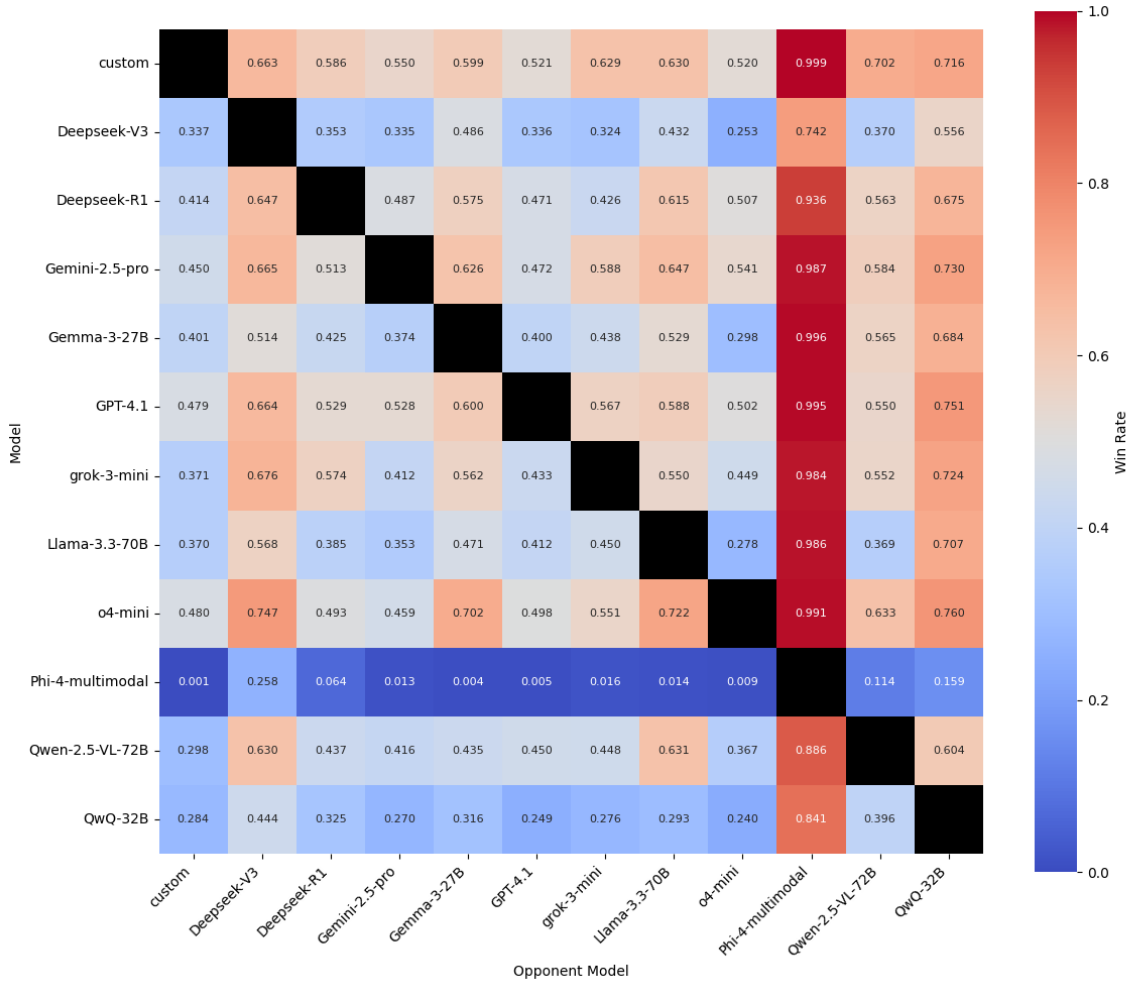

Figure 20: Win rate heatmap for 12 methods in two-player puzzles under the **code-based** setting. Each number represents the win rate of the row entry over the column entry, normalized to ignore ties. For example, for game instances that don't end in a tie, GPT-4.1 beats o4-mini 50.2% of the time and o4-mini beats GPT-4.1 49.8% of the time.