Pencil Puzzle Bench: A Benchmark for Multi-Step Verifiable Reasoning

Justin Waugh
Approximate Labs justin@approximatelabs.com

Abstract

We introduce Pencil Puzzle Bench, a framework for evaluating large language model reasoning through pencil puzzles, a family of constraint-satisfaction problems closely related to NP-complete problems, with deterministic, step-level verification. From a database of 62,231 puzzles across 94 varieties with verified unique solutions, we select a benchmark of 300 puzzles spanning 20 varieties and evaluate 51 models from 11 providers in two modes: *direct ask* (single-shot) and *agentic* (multi-turn with iterative verification). A key differentiator of our benchmark is that **every intermediate board state can be checked against variety-specific constraints**, localizing errors to the exact rule violated, providing the infrastructure for dense, per-move reward signals for process supervision and reinforcement learning.

Our evaluation reveals two distinct axes of capability: (1) *reasoning effort scaling*, where GPT-5.2 improves 81× from no reasoning to maximum effort; and (2) *agentic iteration*, where Claude Opus 4.6 rises from 0.3% to 30.0% through iterative checking, while GPT-5.2@xhigh improves from 20.2% to 56.0%. Agentic attempts span a median of 29 turns over 17 minutes, with the longest exceeding 1,221 turns and 14.3 hours—a demanding test of long-context utilization, not just reasoning.

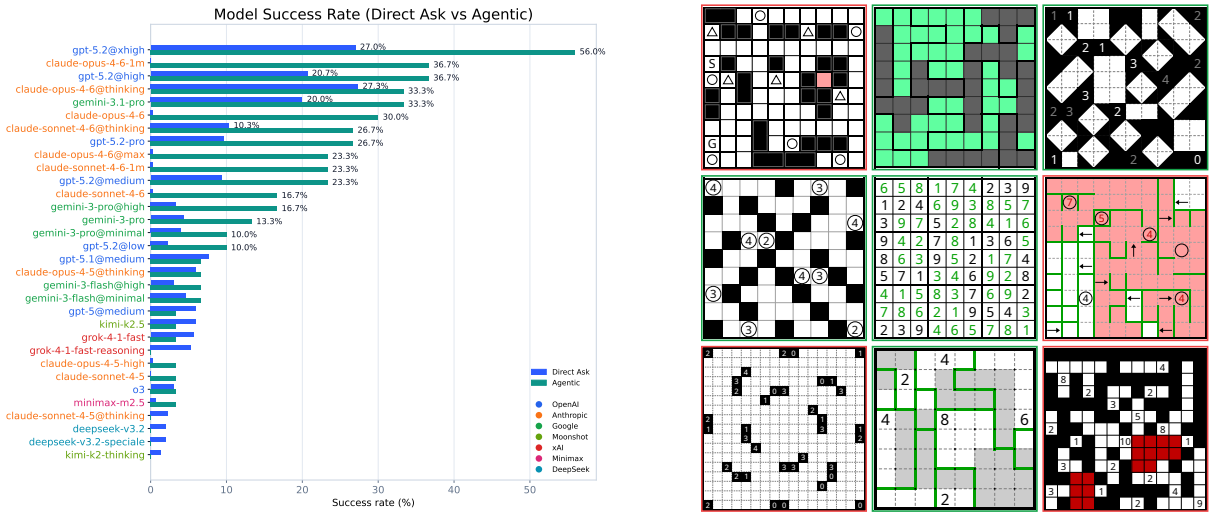


Figure 1: Model success rates by strategy (left), with a 3×3 gallery of partial and complete solves across diverse attempts (right).

1 Introduction

The rapid progress of inference-time compute scaling, from chain-of-thought prompting [40] to dedicated reasoning modes like OpenAI's o-series [26] and DeepSeek-R1 [10], has created urgent demand for benchmarks that can reliably measure multi-step reasoning. Existing benchmarks are either single-turn (GSM8K, MATH, ARC), increasingly contaminated [11, 43], or cannot verify intermediate reasoning steps (SWE-bench checks final code, not reasoning). We need benchmarks that provide **multi-step, grounded, verifiable reasoning with step-level feedback**, enabling process reward models [21, 49], reinforcement learning from verifiable rewards (RLVR) [4], and curriculum learning.

Logic puzzles are an ideal testbed:

- **Verifiable**: Solutions are unambiguously correct or incorrect; each puzzle has exactly one solution verified by a SAT solver
- **Multi-step**: Require chains of deductive reasoning across dozens to hundreds of moves
- **Step-level feedback**: Each move can be validated independently against variety-specific constraints, with error messages identifying *which rule was broken and where*, e.g., "Two shaded cells are adjacent" in nurikabe or "Loop crosses itself" in slitherlink
- **Low contamination risk**: Puzzles are sourced from puzz.link, a Japanese puzzle community. While puzzle *descriptions* may appear in training data as widely-shared URLs, *solutions* are rarely published online, making solution contamination unlikely
- **Order-independent solutions**: The solution board state is unique, but the sequence of moves to reach it is flexible, allowing different orderings, mouse vs keyboard input, and undo/redo without affecting correctness
- **Rich representations**: Boards have ASCII text serialization (what models see), SVG vector rendering, and pixel-rasterized images, enabling future multimodal evaluation

Our contributions:

1. A dataset of 62,231 puzzles across 94 varieties with step-level solution traces, and an evaluation benchmark of 300 puzzles across 20 varieties with programmatic step-level verification, implemented as the `pencil-puzzle-bench` Python package
2. Evaluation of 51 models from 11 providers in two modes (direct ask and agentic), revealing two distinct axes of capability improvement
3. Discovery of the *agentic gap*: agentic iteration benefits models across the capability spectrum, from ++4.8pp for models with extended thinking to ++30.0pp for models without
4. Analysis of reasoning effort scaling, cost-efficiency, and infrastructure limits in agentic evaluation (median 29 turns, up to 1,221 turns per attempt)
5. A difficulty analysis showing that solution compressibility ($R^2_{\text{adj}} = 0.385$) predicts puzzle difficulty 4.2× better than move count ($R^2_{\text{adj}} = 0.092$)

2 Related Work

Reasoning Benchmarks GSM8K [9], MATH [16], and ARC [7] evaluate mathematical and scientific reasoning but are single-turn and have known contamination issues [11, 41, 43]. BIG-Bench [33] and BBH [34] provide broader task coverage. RuleTaker [8] and StrategyQA [12] test logical and implicit reasoning. Recent work has raised concerns about benchmark integrity, evaluation awareness, and the limitations of static benchmarks [5, 24, 32].

Inference-Time Compute and Reasoning Scaling The paradigm of scaling inference-time compute (rather than model size) has emerged as a dominant theme in 2025–2026. OpenAI's o-series models [26] introduced adjustable reasoning effort, DeepSeek-R1 [10] demonstrated that RL can incentivize reasoning capabilities, and subsequent work has explored the frontiers of learning to reason and agentic systems [18]. Complementary inference-time techniques improve reasoning without additional training, including self-consistency [38], least-to-most prompting [51], Plan-and-Solve [37], Program of Thoughts [6], and deliberative search strategies such as Tree of Thoughts [44] and Graph of Thoughts [1] (often paired with self-refinement [23]). Process reward models [21, 49, 50] enable step-level verification for training. Our benchmark directly measures this trend: GPT-5.2's 81× improvement across reasoning effort levels quantifies the returns to inference-time compute on a challenging domain.

Puzzle Benchmarks GridPuzzle [36] and ZebraLogic [22] evaluate logic puzzle solving but focus on text-based grid puzzles without step-level verification. Li et al. [19] study Minesweeper as a reasoning testbed. PuzzleBench [48] provides dynamic multimodal puzzle evaluation. Recent puzzle-centric benchmarks emphasize synthetic verifiability and reflective solving (Enigmata [4], FINEREASON [3]) and challenge models with long-tail logic games (HardcoreLogic [20]); solver-in-the-loop work targets logic puzzle solving in declarative formalisms such as ASP [30]. Giadikiaroglou et al. [13] survey puzzle-solving with LLMs. SATBench [39] generates puzzles from SAT formulas, while RiddleBench [15] tests generative reasoning. Most closely related, Sudoku-Bench [31] evaluates LLMs on 100 Sudoku variant puzzles sourced from Nikoli and competitive puzzle communities, testing "eureka"-moment creative reasoning. Our work is distinguished by breadth (94 puzzle varieties in the full dataset, 20 evaluated, vs. Sudoku variants only), coordinate-based moves with step-level verification against variety-specific constraint sets, and the scale of our agentic evaluation (up to 1,221 turns per attempt).

Tool Use, Agents, and Agentic Benchmarks Early work established the foundations: ReAct [45] combines reasoning with actions, Toolformer [29] teaches models to use tools autonomously, and generative agents [27] demonstrate complex agent behaviors. The field has since produced dedicated agentic benchmarks: SWE-bench [17] evaluates agents on real GitHub issues, tau-bench [46] tests multi-turn tool-agent-user interaction in real-world domains, TheAgentCompany [42] evaluates agents on 175 professional tasks in a simulated company, and the Berkeley Function Calling Leaderboard [28] benchmarks tool-calling capabilities. Recent surveys [14, 18] categorize the rapidly growing space. We contribute a benchmark that evaluates models in both single-shot and agentic modes, with agentic solvers sustaining multi-turn interactions over dozens to hundreds of turns, testing long-context utilization and iterative reasoning on a domain with deterministic step-level verification.

Puzzle Complexity Many pencil puzzles are NP-complete [25, 35, 47], meaning that no efficient general-purpose algorithm is known. While individual instances may admit shortcuts, the theoretical richness of this domain makes it unlikely that surface-level pattern matching suffices across varieties and sizes.

3 The Pencil Puzzle Bench Framework

3.1 Technical Infrastructure

The pzprjs Engine We build on pzprjs[1], a mature JavaScript framework for interactive pencil puzzles that powers the puzz.link community. Pzprjs implements 100+ puzzle varieties with full rule checking, error localization, and completion detection. Puzzles are shared as "classic URLs," compact encodings widely used in the online puzzle community.

Solution Generation Solutions were generated using cspuz-solver2[2], a SAT-based constraint solver that finds verified unique solutions. A key technical contribution is converting from the solver's output format (a complete board state) to the pzprjs move-set format (a sequence of interactive moves), bridging the constraint solver's representation to the puzzle engine's. This conversion enables fine-tuning on solution trajectories, bootstrapping capability in smaller models, analytics on puzzle difficulty, and verification that every puzzle has a unique valid solution. Step-level verification itself comes from the pzprjs `check()` function, which validates each intermediate board state against variety-specific constraints.

Board Representations Each puzzle board supports three representations:

1. **ASCII text serialization**: The format models receive, a structured text encoding of cell contents, borders, and clues
2. **SVG vector rendering**: A scalable vector graphic of the board state, suitable for display and debugging
3. **Pixel images**: SVG can be rasterized to produce actual images, enabling future multimodal evaluation where models "see" the puzzle

3.2 Step-Level Verification

A key contribution of Pencil Puzzle Bench is **step-level verification**: the ability to validate not just final solutions, but every intermediate state during solving. Unlike benchmarks that only report "correct" or "incorrect," our verifier identifies *which specific constraint was violated and where.*

Variety-Specific Constraints Every variety in the dataset defines its own ordered constraint set. Rules are checked in order, with local/direct constraints (e.g., "two shaded cells are adjacent") evaluated before global state constraints (e.g., "all cells must be filled," "loop must be connected"). When only a global rule is broken, no intermediate move is known to violate a local sub-rule. When a direct constraint is broken, the engine provides immediate, localizable error feedback including error highlighting that identifies the offending cells. Examples:

[1] `https://github.com/robx/pzprjs`
[2] `https://github.com/semiexp/cspuz-solver2`

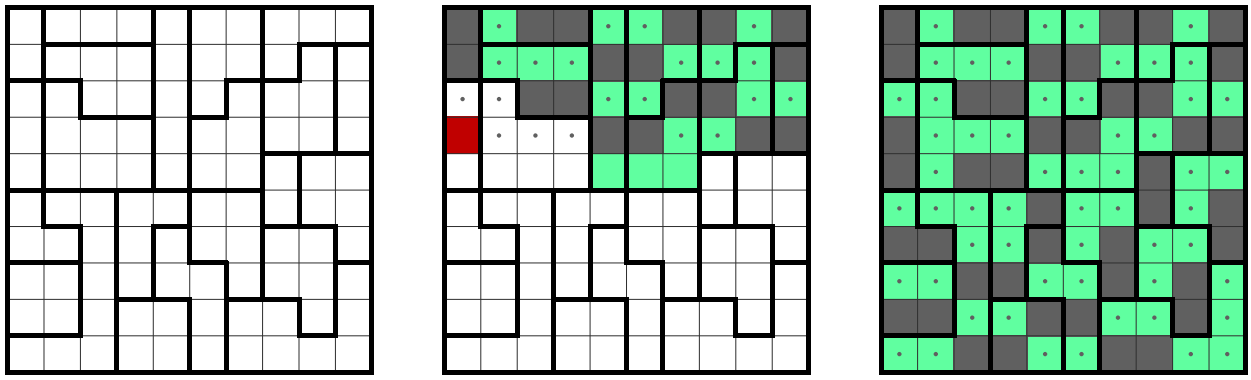
- **Nurikabe**: "Shaded cells form a 2×2 square", "Two numbered islands are connected", "Shaded region is not fully connected"
- **Slitherlink**: "Loop branches at vertex", "Number clue unsatisfied (expected 2, got 3 edges)"
- **Norinori**: "Shaded cell has no adjacent shaded cell", "Region contains fewer than 2 shaded cells"
- **Sudoku**: "Duplicate number in row", "Duplicate number in block"

This per-constraint, per-cell error localization enables models (in agentic mode) to receive targeted feedback about *what went wrong*, not just *that something is wrong*. For RL training, each constraint violation provides the building blocks for dense, interpretable reward signals: the verification infrastructure exposes violation counts, types, and locations at each step, which can be composed into reward functions (e.g., violation-delta shaping, progress-based rewards).

Verification Pipeline Every move in Pencil Puzzle Bench can be validated through a four-step pipeline:

1. **Pre-move state**: Check current board for rule violations
2. **Apply move**: Execute a coordinate-based action (e.g., `mouse,left,3,5`)
3. **Post-move validation**: Immediately check for new violations introduced by the move
4. **Completion check**: Determine if the puzzle is fully and correctly solved

Visualization Figure 2 shows a norinori puzzle at three stages: initial state, partial solution with constraint violations highlighted (red cells indicate which specific rule is broken), and completion.



(a) Initial state (b) Partial solution with errors (c) Completed

Figure 2: A norinori puzzle showing the solving process. Red highlights in (b) indicate cells where variety-specific constraints are violated; the verifier reports which rule is broken (e.g., "Shaded cell has no adjacent shaded cell"), enabling targeted feedback in agentic mode.

3.3 Dataset

Full Database The complete dataset contains **62,231 puzzles** across 94 varieties, sourced from puzz.link, a Japanese puzzle community that aggregates puzzles from Twitter, personal blogs, and puzzle competitions. The dataset includes puzzles from **814 unique creators** spanning multiple years of community activity. Each puzzle has:

- **Verified unique solutions**: Every puzzle was validated via cspuz-solver2 to confirm exactly one solution exists
- **Step-by-step solutions**: Complete move sequences decomposed into *required moves* and *hint moves* (redundant moves that don't change the solution)
- **Source provenance**: Original puzzle URLs, author handles, and publication timestamps

Contamination Pencil puzzles are frequently shared online as puzz.link URLs, but *solutions* are rarely published. The community shares puzzles for others to solve; posting solutions is considered poor etiquette. This means that while puzzle *descriptions* may appear in training data, solution contamination is unlikely even for models trained on web data. This is a stronger contamination argument than mere source obscurity: the cultural norm of the puzzle community actively suppresses solution publication.

Golden Benchmark Set For evaluation, we curated a **300-puzzle golden set**:

- **20 varieties**, selected as the most common varieties with verified solutions
- **15 puzzles per type**, stratified by difficulty:
  - 5 short (fewest required moves within type)
  - 5 medium (middle quintile)
  - 5 long (most required moves within type)

Dataset Summary Table 1 summarizes the key statistics across our dataset tiers.

Table 1: Dataset summary statistics

	Full Dataset	Golden 300	Golden 30	Golden 60
Puzzles	62,231	300	30	60
Varieties	94	20	4	20
Unique creators	814	118	21	–
Median moves	78	54	89	–
Median area	100	100	100	–
SAT-verified	100%	100%	100%	100%

3.4 Python Library

We implemented `pencil-puzzle-bench` as a Python package:

Core API The `Puzzle` class wraps the pzprjs engine with a simple Python interface:

```
from ppbench import Puzzle, load_dataset

puzzle = Puzzle.from_url("http://puzz.link/p?sudoku/...")
puzzle.send_move("mouse,left,3,5")  # Make a move
violations = puzzle.check()          # [] if valid
if puzzle.is_complete():
    print("Solved!")
```

Dataset Loading The golden benchmark set ships with the package:

```
puzzles = load_dataset("golden")     # 300 puzzles
puzzles = load_dataset("golden_30")  # 30-puzzle subset
```

The full 62,231 puzzle dataset includes step-level solutions suitable for Gym-compatible RL training [2].

3.5 Puzzle Varieties

Table 2 shows the 20 varieties in the evaluation set, ordered by solve rate.

Table 2: Puzzle varieties by difficulty (best single-model solve rate across all strategies)

Variety	Best Model	Solve Rate
norinori	gpt-5.2@xhigh	100.0%
shikaku	gpt-5.2@xhigh	86.7%
lits	gpt-5.2@xhigh	73.3%
hitori	claude-opus-4-6@thinking	66.7%
lightup	claude-opus-4-6@thinking	66.7%
mashu	gpt-5.2@xhigh	66.7%
tapa	gpt-5.2@xhigh	66.7%
firefly	gpt-5.2@xhigh	46.7%
nurimisaki	claude-opus-4-6@thinking	40.0%
yajilin	gpt-5.2@xhigh	40.0%
nurikabe	claude-opus-4-6@thinking	33.3%
slither	gpt-5.2@high	33.3%
sudoku	gpt-5.2@xhigh	33.3%
kurodoko	gpt-5.2@xhigh	26.7%
nurimaze	claude-opus-4-6@thinking	26.7%
sashigane*	gpt-5.2@xhigh	20.0%
country	gemini-3.1-pro	13.3%
dbchoco	gpt-5.2@xhigh	6.7%
heyawake*	gpt-5.2@xhigh	6.7%
shakashaka*	gpt-5.2@xhigh	6.7%

*Solved only through agentic iteration (0% in direct-ask evaluation).

4 Evaluation Methodology

4.1 Strategies

Direct Ask (Single-Shot) Model receives puzzle state and must output complete solution as JSON move list. One inference call.

Agentic (Multi-Turn with Iterative Verification) Model has access to tools: `make_move`, `check_board`, `reset_puzzle`. The model iteratively makes moves, checks the board for constraint violations, and course-corrects based on the specific error feedback. This process continues until the puzzle is solved or the model gives up.

Move Format: Moves are coordinate-based strings like `mouse,left,3,5` (left-click at column 3, row 5) or `mouse,right,1,2` (right-click for secondary state). Line-drawing puzzles use drag syntax: `mouse,left,1,1,1,5`. This interface mirrors the puzz.link web UI.

Note: Both strategies are **baselines**, not optimized attempts. We did not perform prompt engineering or hill-climbing. The goal is to measure where models are, not to maximize scores.

4.2 Agentic Evaluation Design

Of 51 models, 3 were excluded from agentic evaluation because they lack the tool-calling capability required by the harness. The remaining 48 were evaluated agentically: 45 on a **30-puzzle baseline** (4 types: yajilin, sashigane, lits, lightup), and 3 top models (GPT-5.2@xhigh, Claude Opus 4.6@thinking, and Gemini 3.1 Pro) on an **expanded 60-puzzle set** across all 20 varieties (3 puzzles per variety). This expansion is what solved the 3 previously unsolved varieties.

Agentic Scale Agentic evaluation is a long-context, many-turn process. Across 1,602 agentic runs (45 models on the 30-puzzle baseline, plus 3 top models on the expanded 60-puzzle set):

- **Turns per attempt**: mean 57.2, median 29, P90 = 113, max 1,221
- **Duration per attempt**: mean 40 min, median 17 min, P90 = 108 min, max 14.3 hours

These are not brief tool calls; models sustain solving attempts over extended multi-turn conversations, maintaining context across dozens to hundreds of reasoning-action-feedback loops. This makes Pencil Puzzle Bench a demanding test of long-context utilization and sustained reasoning, not just single-turn problem solving.

4.3 Models

We evaluate 51 models across 11 providers:

- **OpenAI** (14): GPT-5.2 (none/low/medium/high/xhigh), GPT-5.2-pro, GPT-5.1@medium, GPT-5@medium, GPT-4.1, GPT-4o, o1, o3, GPT-3.5-turbo, GPT-OSS-120B
- **Anthropic** (11): Claude Sonnet 4.5, Claude Sonnet 4.5@thinking, Claude Sonnet 4.6, Claude Sonnet 4.6@thinking, Claude Sonnet 4.6-1m, Claude Opus 4.5 (high), Claude Opus 4.5@thinking, Claude Opus 4.6 (high), Claude Opus 4.6@thinking, Claude Opus 4.6@max, Claude Opus 4.6-1m
- **Google** (7): Gemini 3.1 Pro, Gemini 3 Pro (default/minimal/high), Gemini 3 Flash (minimal/high), Gemini 2.5 Pro

- **xAI** (3): Grok 4.1 Fast, Grok 4.1 Fast Reasoning, Grok Code Fast
- **DeepSeek** (2): V3.2, V3.2-speciale
- **Qwen** (5): Qwen3.5-397B, Qwen3-235B, Qwen3-Coder, Qwen3-Next-80B, Qwen3-VL-235B
- **Mistral** (2): Mistral Large, Devstral
- **Moonshot** (2): Kimi K2@thinking, Kimi K2.5
- **Minimax** (2): M2.1, M2.5
- **Zhipu** (2): GLM-4.7, GLM-5
- **Xiaomi** (1): MiMo-v2

5 Results

5.1 Overall Performance

Figure 1 and Table 3 show the top models ranked by their best success rate across either strategy. Complete results for all 51 models are in Table 6 (Appendix C).

Table 3: Top 15 models by best success rate. Direct ask on 300 puzzles; agentic on 30-puzzle baseline for most models, 60-puzzle expanded set for top 3.

Model	Direct Ask	Agentic
gpt-5.2@xhigh	27.0%	**56.0%**
claude-opus-4-6-1m	0.0%	**36.7%**
gpt-5.2@high	20.7%	**36.7%**
claude-opus-4-6@thinking	27.3%	**33.3%**
gemini-3.1-pro	20.0%	**33.3%**
claude-opus-4-6	0.3%	**30.0%**
claude-sonnet-4-6@thinking	10.3%	**26.7%**
gpt-5.2-pro	9.7%	**26.7%**
claude-opus-4-6@max	0.3%	**23.3%**
claude-sonnet-4-6-1m	0.3%	**23.3%**
gpt-5.2@medium	9.3%	**23.3%**
claude-sonnet-4-6	0.3%	**16.7%**
gemini-3-pro@high	3.3%	**16.7%**
gemini-3-pro	4.3%	**13.3%**
gemini-3-pro@minimal	4.0%	**10.0%**

5.2 The Agentic Gap

The agentic strategy, where models iteratively make moves, check the board for constraint violations, and course-correct, reveals a striking pattern: **models with weak direct-ask performance benefit disproportionately from agentic iteration**. Table 4 shows the top 10 models by agentic success rate with uplift, and Table 5 shows GPT-5.2's performance across reasoning effort levels.

Claude Opus 4.6 at default effort (no extended thinking) achieves 30.0% agentic success despite only 0.3% direct-ask performance (+30.0 percentage points on the 30 puzzles evaluated in both

Table 4: Top 10 models by agentic success rate, with uplift (Δpp = agentic − direct) and cost per attempt.

Model	Direct	Agentic	Δpp	Cost/Attempt
gpt-5.2@xhigh	27.0%	56.0%	+29.0	$9.74
claude-opus-4-6-1m	0.0%	36.7%	+36.7	$14.11
gpt-5.2@high	20.7%	36.7%	+16.0	$7.30
claude-opus-4-6@thinking	27.3%	33.3%	+6.0	$6.24
gemini-3.1-pro	20.0%	33.3%	+13.3	$14.36
claude-opus-4-6	0.3%	30.0%	+29.7	$10.89
claude-sonnet-4-6@thinking	10.3%	26.7%	+16.3	$3.94
gpt-5.2-pro	9.7%	26.7%	+17.0	$41.52
claude-opus-4-6@max	0.3%	23.3%	+23.0	$10.87
claude-sonnet-4-6-1m	0.3%	23.3%	+23.0	$169.27

Table 5: GPT-5.2 performance by reasoning effort level. Direct ask on 300 puzzles; agentic on 30-puzzle baseline.

Model	Direct	Agentic	Uplift
gpt-5.2	0.33%	0.00%	1×
gpt-5.2@low	2.3%	10.0%	7×
gpt-5.2@medium	9.3%	23.3%	28×
gpt-5.2@high	20.7%	36.7%	62×
gpt-5.2@xhigh	27.0%	56.0%	81×

modes). This is the most extreme case: the model lacks internal chain-of-thought reasoning, so agentic iteration substantially compensates. Even with extended thinking enabled, the gap persists: Claude Opus 4.6@thinking scores 27.3% direct-ask but still gains +4.8pp from agentic iteration (28.6% → 33.3% on the same 84 puzzles). GPT-5.2@xhigh, which already achieves 27.0% direct-ask, gains +35.7pp from agentic iteration (20.2% → 56.0% on the same 84 puzzles). This suggests that reasoning depth and agentic iteration are **two distinct axes of capability**: the agentic gap ranges from ++4.8pp for models with strong reasoning to ++30.0pp for models without, and both benefit from iterative verification and course-correction.

Extended Agentic Evaluation The initial 30-puzzle agentic evaluation covered 4 varieties. We then ran an expanded 60-puzzle evaluation across all 20 varieties for the top 3 models (GPT-5.2@xhigh, Claude Opus 4.6@thinking, and Gemini 3.1 Pro). Infrastructure error rates (timeouts, API failures) differed between models: GPT-5.2@xhigh had a 27.8% error rate on the baseline set vs. 39.6% on the expanded puzzles, while Claude Opus 4.6@thinking showed 50.0% and 31.2% respectively, suggesting different failure modes across models.

Claude's Arc At the same configuration (default effort, no extended thinking), agentic performance improves across Claude generations: Sonnet 4.5 (0.0% / 3.3%), Opus 4.5 (0.3% / 3.3%), Opus 4.6 (0.3% / 30.0%). Without extended thinking, the jump from Opus 4.5 to 4.6 is entirely in agentic mode. With extended thinking enabled, both axes improve: Opus 4.5@thinking (6.0% / 6.7%) to Opus 4.6@thinking (27.3% / 33.3%).

Higher Effort Does Not Always Help Among non-thinking configurations, Claude Opus 4.6 at default effort (high) outperforms Opus 4.6@max (30.0% vs 23.3% on the baseline 30-puzzle set). Enabling extended thinking (Opus 4.6@thinking, 33.3% on the expanded set) does not surpass the default effort configuration on the baseline puzzles, suggesting that the relationship between reasoning depth and agentic effectiveness is non-monotonic.

Agentic Iteration Solves Previously Unsolved Varieties The expanded agentic evaluation solved 3 previously unsolved varieties; all 20 varieties have now been solved at least once. This demonstrates that iterative verification and course-correction can unlock capabilities not achieved by any model in single-shot mode.

5.3 Reasoning Effort Scaling

Figure 3 shows GPT-5.2's outcome breakdown across reasoning effort levels. Each attempt has one of three outcomes: *correct* (puzzle solved), *incorrect* (model returned a response but the solution was wrong), or *request failed* (infrastructure timeout or API error before any response was returned).

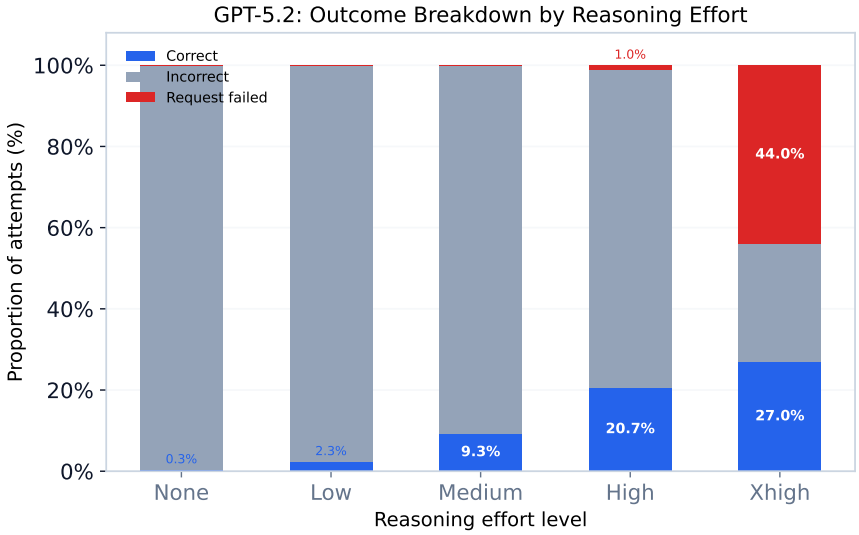


Figure 3: GPT-5.2 outcome breakdown by reasoning effort level (direct ask, 300 puzzles). Each bar sums to 100%. Correct answers (blue) improve 81× from none to xhigh, but at xhigh, 35% of requests fail before returning a response (red), revealing a sharp reliability/capability tradeoff.

5.4 Model Capability Over Time

Figure 4 shows model release dates plotted against success rates, illustrating the pace of capability improvement. At medium reasoning effort, we observe steady improvement across GPT-5 generations: GPT-5.0 (6.0%), GPT-5.1 (7.7%), GPT-5.2 (9.3%). A breakaway in capability is visible since late 2025, with dramatic improvements in the most recent three months. However, even the best models reach only 27.0% success in direct-ask mode, and many varieties and larger puzzles remain unsolved across all models. The extended timeline in Figure 7 (Appendix E) shows that models released before late 2024 achieve 0% or near-0% solve rates, confirming that this capability is entirely new.

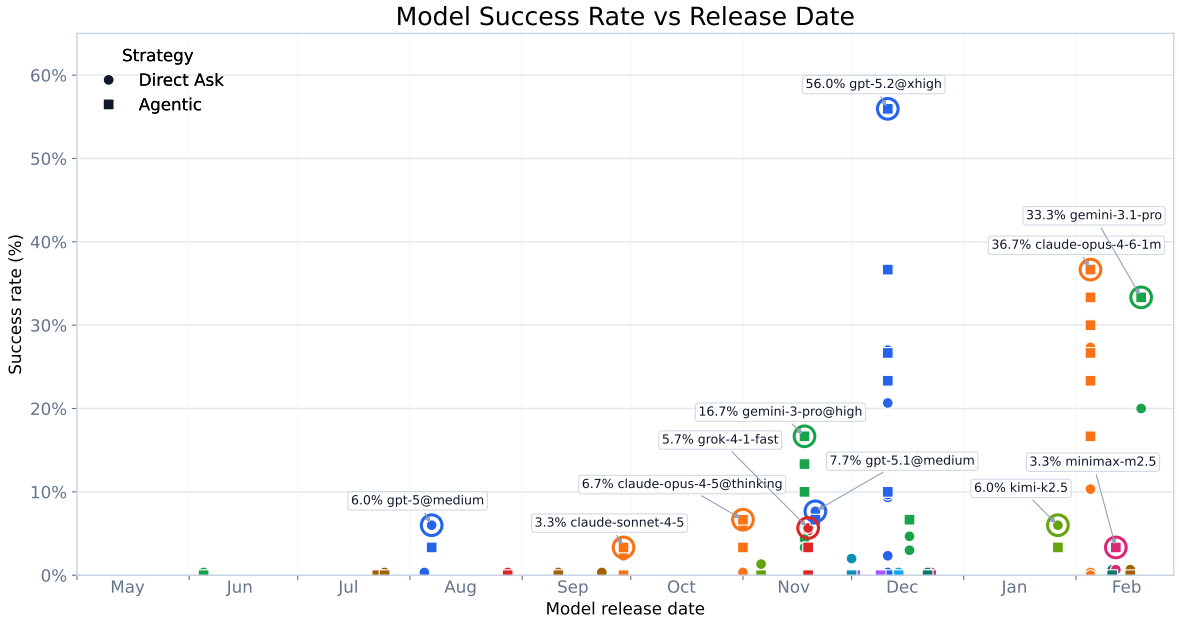


Figure 4: Model success rates over time with frontier model release dates annotated. The progression shows both generational improvement within model families and the gap between frontier and non-frontier models.

5.5 Cost Analysis

Total recorded benchmark cost: approximately $28,246 across 17,032 runs, computed from per-request token usage in the published artifacts.[3] All run records (model, puzzle, strategy, outcome, cost, duration) are published in the `runs.jsonl` artifact on HuggingFace.[4] Figure 5 visualizes the cost-success Pareto frontier.

Cost-per-success varies by **66,822**× across models.

6 Analysis

6.1 What Makes a Puzzle Hard?

Our benchmark stratifies puzzles into short, medium, and long tiers by *number of required moves* within each type. But is move count a good proxy for difficulty? After analysis, we propose **solution compression ratio** as a better difficulty measure.

Move Count is a Weak Predictor Regressing solve rate on move count yields $R^2_{\text{adj}} = 0.092$; move count explains only 0.092 of variance in solve rate. Puzzle variety alone ($\eta^2 = 0.242$) explains 2.6× more variance than move count.

Solution Compressibility Predicts Difficulty The **compression ratio** of the solution move sequence (zlib-compressed size / raw size) achieves $R^2_{\text{adj}} = 0.385$, 4.2× more predictive than move

[3]These are artifact-recorded costs. Actual provider costs are higher than recorded because when agentic runs terminate with errors, token usage from completed turns before the error is not recorded.

[4]`https://huggingface.co/datasets/approximatelabs/pencil-puzzle-bench`

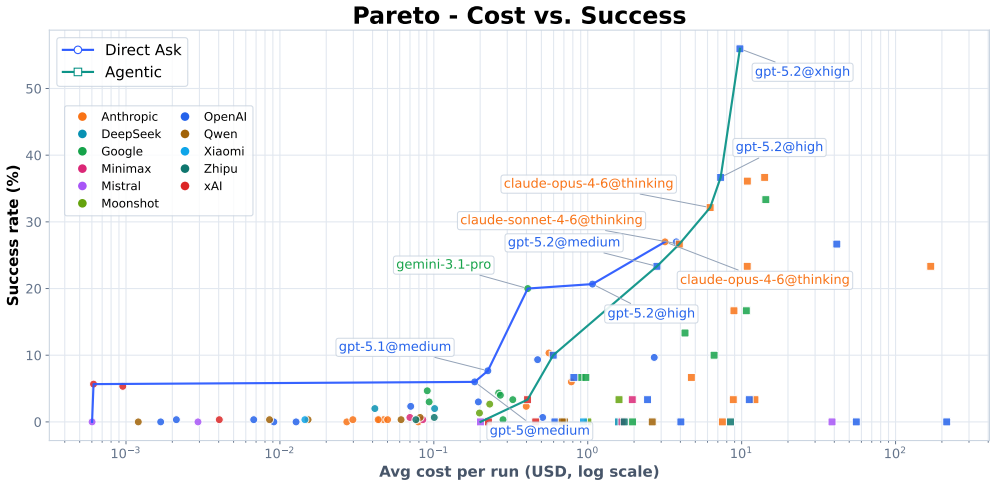


Figure 5: Cost per success vs. success rate across models. The Pareto frontier (lower-right) shows the cost-efficiency tradeoff: Grok 4.1 Fast achieves the lowest cost per success ($0.01) while GPT-5.2@xhigh achieves the highest success rate (56.0%).

count. The intuition: solutions with high compression ratios contain repetitive, structured patterns that models can generalize from partial progress. Low compression ratios indicate high-entropy solutions requiring genuinely novel decisions at each step.

Why Area Doesn't Add Much Adding grid area improves the model only modestly (compression + area + log(area): $R^2_{\text{adj}} = 0.415$ vs. compression alone: 0.385). This is likely because area is confounded with variety; different varieties tend to use characteristic grid sizes. Within a variety, puzzles often cluster in a narrow size range (the Golden 300 has a median area of 100 for nearly every variety), so area provides little additional signal once variety is controlled for. Adding variety dummies to the best three features yields $R^2_{\text{adj}} = 0.621$, confirming that variety captures additional difficulty structure beyond compressibility.

6.2 Infrastructure Scaling Limits

The 35% request failure rate at xhigh reasoning effort is not random network failure. Error duration analysis reveals 100% of failures occur between 1–4 hours, with 62% clustering in the 2–3 hour range, suggesting a systematic server-side timeout around 2.5–3 hours. These are not transient errors but systematic limits on how long a single inference request can run.

The tradeoff is stark (Figure 3): xhigh gains +6pp over high in direct ask (20.7% → 27.0%), with corresponding gains in agentic mode (36.7% → 56.0%). But at xhigh, 35% of requests fail before returning any response. The marginal capability gain is real, but reliability collapses.

7 Discussion

Two Axes of Capability Our results reveal two distinct axes along which models can improve at puzzle solving: *reasoning depth* (controlled by effort levels and extended thinking) and *agentic iteration* (controlled by iterative verification and course-correction). These axes are complementary, not redundant. The agentic gap is largest for models without internal reasoning: Claude Opus 4.6 (no extended thinking) has near-zero direct-ask performance but reaches 30.0% agentically. With

extended thinking enabled, Opus 4.6@thinking achieves 27.3% direct-ask—yet still gains +4.8pp from agentic iteration (28.6% → 33.3% on the same 84 puzzles). GPT-5.2@xhigh, the strongest single-shot reasoner at 27.0%, gains a further ++35.7pp agentically (56.0%). The strongest results combine both axes: deep reasoning *and* iterative verification. The expanded agentic evaluation solved 3 previously unsolved varieties that no model solved in direct-ask mode, demonstrating that agentic iteration provides capabilities beyond what single-shot reasoning alone achieves.

A Grounding Example To contextualize these results: the best model at maximum reasoning effort (GPT-5.2@xhigh) solves only 33.3% of Sudoku puzzles, a variety familiar to most readers. This underscores that even well-known constraint-satisfaction problems remain genuinely challenging for frontier models, despite Sudoku being one of the most commonly encountered varieties in training data.

Limitations Models received ASCII text-based board representations. In agentic mode, models also had access to a `render_board_as_svg()` tool that returns an SVG text representation of the current board state, which some models called for additional spatial context. Rasterized pixel images were not used. Varieties requiring strong visual-spatial reasoning may benefit from multimodal approaches using rendered images. The agentic evaluation used a **30-puzzle subset** (4 varieties: yajilin, sashigane, lits, lightup, selected to span a range of difficulty levels and elicit meaningful agentic iteration) for most models, with an expanded **60-puzzle evaluation** (3 puzzles per variety across all 20 varieties, selected by difficulty tier) for the top 3 models. Both strategies are baselines without prompt optimization.

All success rates are point estimates from single evaluation runs without confidence intervals or repeated trials. With 15 puzzles per variety, a single additional solve changes per-variety rates by 6.7 percentage points. The agentic uplift figures are computed from small samples and should be interpreted with this caveat. No human performance baseline is established. Future work should include repeated trials, bootstrap confidence intervals, and human calibration studies.

Future Work The puzzle dataset, benchmark code, and step-level solution traces enable:

- RL fine-tuning with step-level rewards, following the paradigm of process supervision [21, 49] and reasoning via RL [10]
- Process reward model training using per-move, per-constraint verification signals
- Curriculum learning across difficulty levels using compression ratio as a principled difficulty metric
- Multimodal evaluation using the SVG renderer, where models could "see" the puzzle board as a rendered image rather than parsing ASCII text

8 Conclusion

We introduced Pencil Puzzle Bench, a dataset of 62,231 logic puzzles across 94 varieties with step-level solution traces, and an evaluation benchmark of 300 puzzles across 20 varieties with programmatic step-level verification, where every intermediate board state can be checked against variety-specific constraints. Our evaluation of 51 models in two modes (direct ask and agentic) reveals two distinct axes of improvement: reasoning effort scaling (81× from GPT-5.2 none to xhigh) and agentic

iteration (up to ++30.0pp for models without extended thinking, and ++4.8pp even for strong reasoning models). The strongest results combine both axes: GPT-5.2@xhigh achieves 56.0% in agentic mode. Agentic evaluation solved 3 previously unsolved varieties; all 20 varieties have now been solved at least once, though 49% of individual puzzles remain unsolved by any model.

The step-level verification infrastructure provides a foundation for future work on process supervision, step-level reinforcement learning, and curriculum learning for reasoning.

Acknowledgments This paper was written with heavy use of AI coding agents (Claude Code, Codex) and conversational AI assistants (ChatGPT, Claude). These tools were used extensively throughout the research, implementation, data analysis, and writing process. Any errors or inaccuracies in this work are the sole responsibility of the author.

References

[1] M. Besta, N. Blach, et al. Graph of thoughts: Solving elaborate problems with large language models, 2024. URL https://arxiv.org/abs/2308.09687.

[2] G. Brockman, V. Cheung, et al. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

[3] G. Chen, W. Xu, et al. Finereason: Evaluating and improving llms' deliberate reasoning through reflective puzzle solving, 2025. URL https://arxiv.org/abs/2502.20238.

[4] J. Chen, Q. He, et al. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles, 2025. URL https://arxiv.org/abs/2505.19914.

[5] S. Chen, Y. Chen, et al. Recent advances in large language model benchmarks against data contamination: From static to dynamic evaluation, 2025. URL https://arxiv.org/abs/2502.17521.

[6] W. Chen, X. Ma, et al. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023. URL https://arxiv.org/abs/2211.12588.

[7] P. Clark, I. Cowhey, et al. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

[8] P. Clark, O. Tafjord, and K. Richardson. Transformers as soft reasoners over language, 2020. URL https://arxiv.org/abs/2002.05867.

[9] K. Cobbe, V. Kosaraju, et al. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

[10] DeepSeek-AI, D. Guo, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2026. URL https://arxiv.org/abs/2501.12948.

[11] C. Deng, Y. Zhao, et al. Investigating data contamination in modern benchmarks for large language models, 2024. URL https://arxiv.org/abs/2311.09783.

[12] M. Geva, D. Khashabi, et al. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies, 2021. URL https://arxiv.org/abs/2101.02235.

[13] P. Giadikiaroglou, M. Lymperaiou, et al. Puzzle solving using reasoning of large language models: A survey, 2024. URL https://arxiv.org/abs/2402.11291.

[14] T. Guo, X. Chen, et al. Large language model based multi-agents: A survey of progress and challenges, 2024. URL https://arxiv.org/abs/2402.01680.

[15] D. Halder, A. Saji, et al. Riddlebench: A new generative reasoning benchmark for llms, 2025. URL https://arxiv.org/abs/2510.24932.

[16] D. Hendrycks, C. Burns, et al. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

[17] C. E. Jimenez, J. Yang, et al. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

[18] Z. Ke, F. Jiao, et al. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems, 2025. URL https://arxiv.org/abs/2504.09037.

[19] Y. Li, H. Wang, and C. Zhang. Assessing logical puzzle solving in large language models: Insights from a minesweeper case study, 2024. URL https://arxiv.org/abs/2311.07387.

[20] J. Liang, S. Wan, et al. Hardcorelogic: Challenging large reasoning models with long-tail logic puzzle games, 2025. URL https://arxiv.org/abs/2510.12563.

[21] H. Lightman, V. Kosaraju, et al. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

[22] B. Y. Lin, R. L. Bras, et al. Zebralogic: On the scaling limits of llms for logical reasoning, 2025. URL https://arxiv.org/abs/2502.01100.

[23] A. Madaan, N. Tandon, et al. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

[24] T. R. McIntosh, T. Susnjak, et al. Inadequacies of large language model benchmarks in the era of generative artificial intelligence, 2024. URL https://arxiv.org/abs/2402.09880.

[25] A. V. Nikolaev. Evolomino is np-complete, 2025. URL https://arxiv.org/abs/2503.07611.

[26] OpenAI, :, et al. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

[27] J. S. Park, J. C. O'Brien, et al. Generative agents: Interactive simulacra of human behavior, 2023. URL https://arxiv.org/abs/2304.03442.

[28] S. G. Patil, H. Mao, et al. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In A. Singh, M. Fazel, et al., editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

[29] T. Schick, J. Dwivedi-Yu, et al. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

[30] T. P. Schrader, L. Lange, et al. A solver-in-the-loop framework for improving llms on answer set programming for logic puzzle solving, 2025. URL https://arxiv.org/abs/2512.17093.

[31] J. Seely, Y. Imajuku, et al. Sudoku-bench: Evaluating creative reasoning with sudoku variants, 2025. URL https://arxiv.org/abs/2505.16135.

[32] Z. Shi, X. Jiang, et al. Judgeagent: Beyond static benchmarks for knowledge-driven and dynamic llm evaluation, 2026. URL https://arxiv.org/abs/2509.02097.

[33] A. Srivastava, A. Rastogi, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023. URL https://arxiv.org/abs/2206.04615.

[34] M. Suzgun, N. Scales, et al. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

[35] H. Tang. A framework for loop and path puzzle satisfiability np-hardness results, 2022. URL https://arxiv.org/abs/2202.02046.

[36] N. Tyagi, M. Parmar, et al. Step-by-step reasoning to solve grid puzzles: Where do llms falter?, 2024. URL https://arxiv.org/abs/2407.14790.

[37] L. Wang, W. Xu, et al. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023. URL https://arxiv.org/abs/2305.04091.

[38] X. Wang, J. Wei, et al. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

[39] A. Wei, Y. Wu, et al. Satbench: Benchmarking llms' logical reasoning via automated puzzle generation from sat formulas, 2025. URL https://arxiv.org/abs/2505.14615.

[40] J. Wei, X. Wang, et al. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903v6.

[41] C. Xu, S. Guan, et al. Benchmark data contamination of large language models: A survey, 2024. URL https://arxiv.org/abs/2406.04244.

[42] F. F. Xu, Y. Song, et al. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2025. URL https://arxiv.org/abs/2412.14161.

[43] R. Xu, Z. Wang, et al. Benchmarking benchmark leakage in large language models, 2024. URL https://arxiv.org/abs/2404.18824.

[44] S. Yao, D. Yu, et al. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

[45] S. Yao, J. Zhao, et al. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

[46] S. Yao, N. Shinn, et al. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

[47] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. Technical report, Nov 2002. URL https://ipsj.ixsq.nii.ac.jp/records/31947.

[48] Z. Zhang, Z. Chen, et al. Puzzlebench: A fully dynamic evaluation framework for large multimodal models on puzzle solving, 2025. URL https://arxiv.org/abs/2504.10885.

[49] Z. Zhang, C. Zheng, et al. The lessons of developing process reward models in mathematical reasoning, 2025. URL https://arxiv.org/abs/2501.07301.

[50] C. Zheng, J. Zhu, et al. A survey of process reward models: From outcome signals to process supervisions for large language models, 2025. URL https://arxiv.org/abs/2510.08049.

[51] D. Zhou, N. Schärli, et al. Least-to-most prompting enables complex reasoning in large language models, 2023. URL https://arxiv.org/abs/2205.10625v3.

Appendix A: Puzzle Variety Descriptions

Brief descriptions of the 20 varieties in the evaluation set. The full dataset contains 94 varieties.

Country Draw a loop through orthogonally adjacent cells that visits every outlined country exactly once. Numbers indicate how many cells the loop passes through in that country. The loop cannot branch or cross itself.

Dbchoco Divide the grid into regions. Each region contains one white and one grey contiguous area of identical size and shape (allowing rotation/reflection). Numbers indicate the size of the area they're in.

Firefly Draw paths from every firefly (starting at black dots) to form one connected network. Paths cannot branch, cross, or connect directly between two black dots. Numbers indicate how many times the path turns.

Heyawake Shade some cells in a grid divided into rooms. Shaded cells cannot be adjacent. Numbers indicate shaded cells in that room. No horizontal/vertical line of unshaded cells can pass through 2+ room borders.

Hitori Shade some cells so that no row or column contains duplicate unshaded numbers. Shaded cells cannot be adjacent. All unshaded cells must form one orthogonally connected area.

Kurodoko Shade some cells (not numbers). Shaded cells cannot be adjacent. Numbers indicate how many unshaded cells can be seen in a straight line horizontally and vertically, including itself. All unshaded cells must connect.

Lightup Place lights in empty cells to illuminate all non-black cells. Lights illuminate in straight lines until blocked by black cells. Lights cannot illuminate each other. Numbers on black cells indicate adjacent lights.

Lits Place one tetromino (4-cell block) in every outlined region. No 2x2 square can be fully covered. Identical tetrominoes (including rotations/reflections) cannot share an edge. All tetrominoes must connect.

Mashu Draw a loop through every circle. On black circles: the loop must turn and go straight before/after. On white circles: the loop must go straight but turn in at least one adjacent cell.

Norinori Shade cells so that each shaded cell is adjacent to exactly one other shaded cell (forming dominoes). Each outlined region contains exactly 2 shaded cells.

Nurikabe Shade cells to form regions of unshaded cells. Each region contains exactly one number indicating its size. Numbers cannot be shaded. Shaded cells cannot form 2x2 squares and must all connect.

Nurimaze Shade some tiles to form a maze. Tiles are fully shaded or unshaded. Clued tiles cannot be shaded. No 2x2 square can be all shaded or all unshaded. Unshaded cells form a path from S to G passing through all circles.

Nurimisaki Shade cells with no 2x2 squares of same color. Circles mark cells that are unshaded and adjacent to exactly one other unshaded cell. Numbers indicate visible unshaded cells in a straight line including itself.

Sashigane Divide the grid into L-shaped regions (width one cell). Circles must be at the corner of an L. Arrows must be at the ends, pointing toward the corner. Numbers indicate total cells in that L-shape.

Shakashaka Place right triangles (half-cells) so that every unshaded area forms a rectangle (upright or 45° rotated). Numbers indicate how many adjacent cells contain triangles.

Shikaku Divide the grid into rectangles. Each rectangle contains exactly one number indicating its size in cells.

Slither Draw a single loop along cell edges. Numbers indicate how many of that cell's four edges are part of the loop. The loop cannot branch or cross.

Sudoku Place numbers 1-N in each cell (N = grid width). Each row, column, and outlined block contains exactly one of each number.

Tapa Shade cells (not numbers). Numbers indicate lengths of consecutive shaded cell blocks in the 8 surrounding cells. Shaded cells cannot form 2x2 squares and must all connect.

Yajilin Shade some cells and draw a loop through the rest. Shaded cells cannot be adjacent. Number clues cannot be shaded and indicate how many shaded cells lie in the arrow's direction.

Appendix B: Prompt Templates

B.1 DirectAsk (Single-Shot) Strategy

System Prompt:

````
Solve the puzzle!!

Answer with a list of moves you would like to make that
solve the puzzle as json in a markdown json code block

```json
["mouse,left,1,1", "mouse,right,3,1", ... ]
```
````

User Prompt Template:

```
Puzzle Type: {puzzle_type}
Puzzle Rules:
```
{rules_text}
```

Here is an example of inputs / a solved puzzle
  (lots of context for you)
```
{example_of_inputs}
```
Here's some more:
{example_move_context}
Note specifically how the coordinate systems work
  (for the puzzle vs. the inputs).
For the puzzle you are working on, ensure you fully
  understand from the example of input above, that
  the move is exactly where you expect.
==== ==== ==== ==== ====
Here is the puzzle you are to solve:
{puzzle_state}

==== ==== ==== ==== ====
Please now solve it.
```

B.2 BasicAgentic (Multi-Turn) Strategy

System Prompt:

```
Solve the puzzle!!
It is known to be solvable, and you can figure it out.
This is a logic deduction benchmark.
You are graded on both how many steps you take (number of
tool calls), how many moves it takes (puzzle moves), and
ultimately if you can solve the puzzle. This puzzle is
solvable. If you need to, you may reset the puzzle and
keep trying.
```

User Prompt: Same template as DirectAsk (Section B.1).
Available Tools:

- `make_move(movestring)`: "Make a move, shows the board after the move is applied"
- `make_multi_move(movelist)`: "Make a series of moves, shows the board after the move is applied"
- `check_board_for_completeness()`: "Check the current state of the board against the rules of the puzzle, see if its complete or if errors exist"
- `render_board_as_svg()`: "Shows the full detail SVG of the board (useful if you want more information / are worried about your view into errors)"
- `get_rules()`: "Gets all the rules for the puzzle"

- `reset_puzzle()`: "Fully reset the puzzle (erase all moves, go back to a blank slate). Use this instead of giving up if you want another attempt."
- `give_up()`: Forfeit the attempt

Output Validation Loop: If the model produces a text response before the puzzle is complete, the agent framework automatically retries with the message: "Not done yet, keep going!! Puzzle isn't complete." This continues until the puzzle is solved, the model calls `give_up()`, or the maximum move limit (5000) is reached.

Appendix C: Full Model Results

Complete results for all 51 evaluated models across both strategies.

Appendix D: Puzzle Variety Gallery

Figure 6 shows one puzzle from each of the 20 benchmark varieties, each partially solved. Red highlighting indicates cells where variety-specific constraints are currently violated.

Appendix E: Full Model Success Over Time

Figure 7 extends the timeline from the main text (Figure 4) to include all evaluated models, spanning release dates from 2022 through early 2026. The extended view reveals that **pencil puzzle solving capability is entirely a recent phenomenon**: models released before late 2024—including GPT-3.5-turbo, GPT-4o, GPT-4.1, and o1—achieve 0% or near-0% solve rates. The first non-trivial performance appeared with o3 (3.0%) in early 2025, followed by an explosion in capability starting around November 2025 with the GPT-5 family and reasoning-enabled models. This back-testing confirms that our benchmark measures a genuinely new capability frontier, not one that was latent in earlier model generations.

Table 6: Full results by model and strategy. Succ = correct solves, Fail = wrong answers, Err = infrastructure failures (timeouts, API errors), $/Att = cost per attempt. Strategies are evaluated on different puzzle sets and rates are **not** comparable across columns.

	Direct Ask (300 puzzles)					Agentic (30/60 puzzles)				
Model	Succ	Fail	Err	Rate	$/Att	Succ	Fail	Err	Rate	$/Att
gpt-5.2@xhigh	81	87	132	27.0%	$3.76	47	8	29	**56.0%**	$9.74
claude-opus-4-6-1m	0	300	0	0.0%	$0.08	11	19	0	**36.7%**	$14.11
gpt-5.2@high	62	235	3	20.7%	$1.07	11	18	1	**36.7%**	$7.30
claude-opus-4-6@thinking	82	207	11	27.3%	$3.18	28	23	33	**33.3%**	$6.24
gemini-3.1-pro	60	240	0	20.0%	$0.41	28	47	9	**33.3%**	$14.36
claude-opus-4-6	1	299	0	0.3%	$0.05	9	1	20	**30.0%**	$10.89
claude-sonnet-4-6@thinking	31	263	6	10.3%	$0.56	8	0	22	**26.7%**	$3.94
gpt-5.2-pro	29	137	134	9.7%	$2.70	8	12	10	**26.7%**	$41.52
claude-opus-4-6@max	1	299	0	0.3%	$0.04	7	5	18	**23.3%**	$10.87
claude-sonnet-4-6-1m	1	299	0	0.3%	$0.05	7	3	20	**23.3%**	$169.27
gpt-5.2@medium	28	272	0	9.3%	$0.47	7	23	0	**23.3%**	$2.81
claude-sonnet-4-6	1	299	0	0.3%	$0.03	5	0	25	**16.7%**	$8.91
gemini-3-pro@high	10	290	0	3.3%	$0.33	5	25	0	**16.7%**	$10.72
gemini-3-pro	13	287	0	4.3%	$0.26	4	26	0	**13.3%**	$4.29
gemini-3-pro@minimal	12	287	1	4.0%	$0.27	3	27	0	**10.0%**	$6.62
gpt-5.2@low	7	293	0	2.3%	$0.07	3	27	0	**10.0%**	$0.60
gpt-5.1@medium	23	277	0	**7.7%**	$0.22	2	28	0	6.7%	$0.81
claude-opus-4-5@thinking	18	272	10	6.0%	$0.78	2	26	2	**6.7%**	$4.71
gemini-3-flash@high	9	291	0	3.0%	$0.09	2	28	0	**6.7%**	$0.90
gemini-3-flash@minimal	14	285	1	4.7%	$0.09	2	27	1	**6.7%**	$0.97
gpt-5@medium	18	282	0	**6.0%**	$0.18	1	29	0	3.3%	$11.21
kimi-k2.5	18	282	0	**6.0%**	$0.23	1	0	29	3.3%	$1.60
grok-4-1-fast	17	283	0	**5.7%**	$0.001	1	19	10	3.3%	$0.41
grok-4-1-fast-reasoning	16	283	1	**5.3%**	$0.001	0	18	12	0.0%	$0.46
claude-opus-4-5-high	1	299	0	0.3%	$0.04	1	25	4	**3.3%**	$8.82
claude-sonnet-4-5	0	300	0	0.0%	$0.03	1	10	19	**3.3%**	$12.20
minimax-m2.5	2	237	61	0.7%	$0.07	1	17	12	**3.3%**	$1.95
o3	9	291	0	3.0%	$0.19	1	29	0	**3.3%**	$2.45
claude-sonnet-4-5@thinking	7	293	0	**2.3%**	$0.40	0	16	14	0.0%	$7.49
deepseek-v3.2	6	289	5	**2.0%**	$0.04	0	27	3	0.0%	$1.58
deepseek-v3.2-speciale	6	233	61	**2.0%**	$0.10	–	–	–	–	–
kimi-k2-thinking	4	296	0	**1.3%**	$0.20	0	23	7	0.0%	$1.00
glm-5	2	298	0	**0.7%**	$0.10	0	28	2	0.0%	$8.46
o1	2	298	0	**0.7%**	$0.51	0	30	0	0.0%	$4.03
qwen3.5-397b-a17b	2	296	2	**0.7%**	$0.08	0	0	30	0.0%	$0.000
gemini-2.5-pro	1	299	0	**0.3%**	$0.28	0	30	0	0.0%	$1.96
glm-4.7	1	299	0	**0.3%**	$0.08	0	29	1	0.0%	$1.73
gpt-5.2	1	299	0	**0.3%**	$0.007	0	30	0	0.0%	$0.61
gpt-oss-120b	1	299	0	**0.3%**	$0.002	–	–	–	–	–
grok-code-fast-1	1	297	2	**0.3%**	$0.004	0	17	13	0.0%	$0.23
mimo-v2-flash	1	296	3	**0.3%**	$0.01	0	22	8	0.0%	$0.94
minimax-m2.1	1	299	0	**0.3%**	$0.08	0	24	6	0.0%	$1.67
qwen3-235b-a22b-thinking-2507	1	299	0	**0.3%**	$0.02	0	29	1	0.0%	$0.71
qwen3-next-80b-a3b-thinking	1	298	1	**0.3%**	$0.009	0	8	22	0.0%	$2.62
qwen3-vl-235b-a22b-thinking	1	294	5	**0.3%**	$0.06	–	–	–	–	–
devstral-2512	0	300	0	0.0%	$0.001	0	25	5	0.0%	$0.20
gpt-3.5-turbo	0	300	0	0.0%	$0.002	0	0	30	0.0%	$0.000
gpt-4.1	0	300	0	0.0%	$0.009	0	19	11	0.0%	$214.97
gpt-4o	0	300	0	0.0%	$0.01	0	22	8	0.0%	$55.57
mistral-large-2512	0	300	0	0.0%	$0.003	0	26	4	0.0%	$38.65
qwen3-coder	0	300	0	0.0%	$0.001	0	30	0	0.0%	$0.68

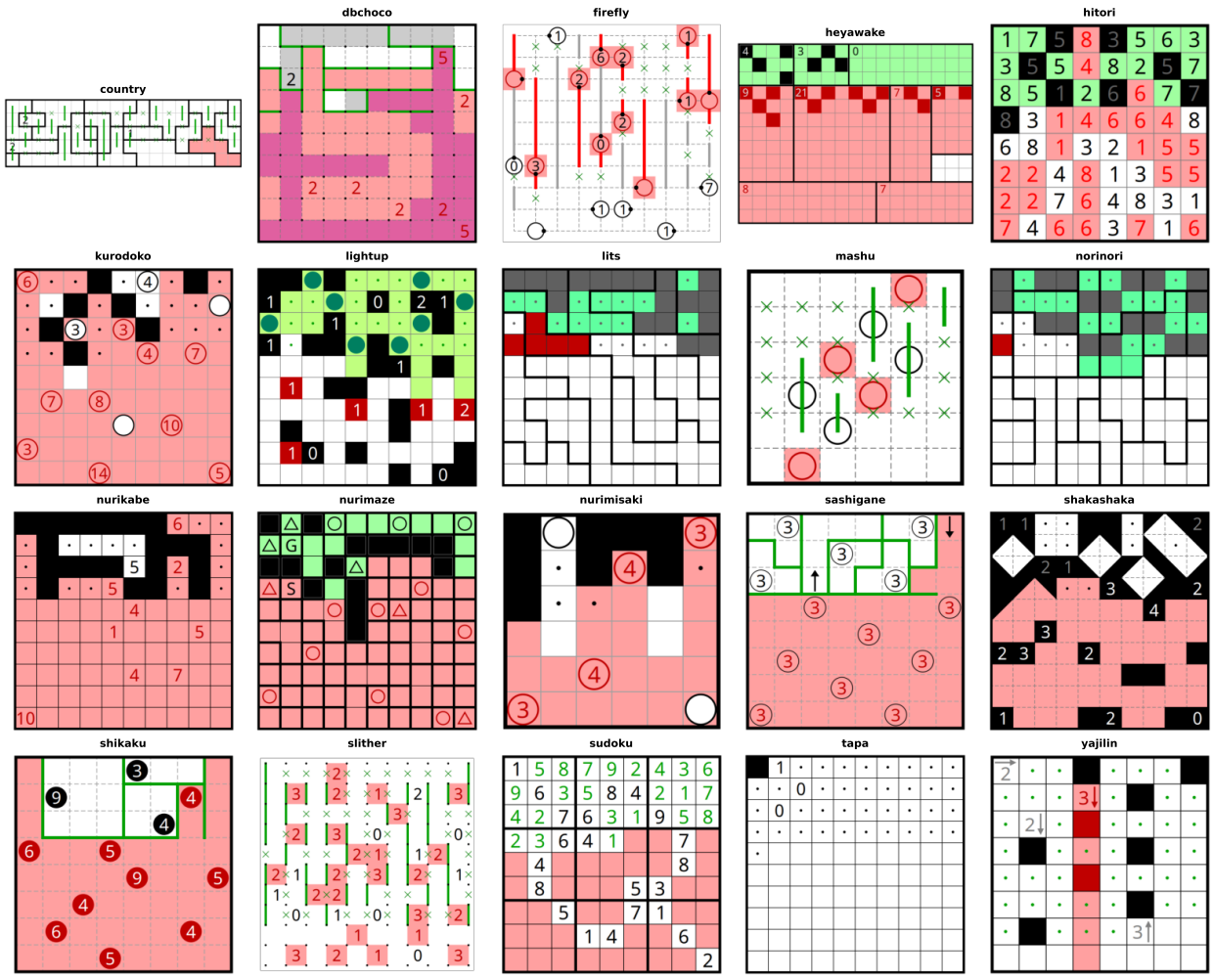


Figure 6: All 20 varieties in the benchmark, shown mid-solve with constraint violations highlighted in red.

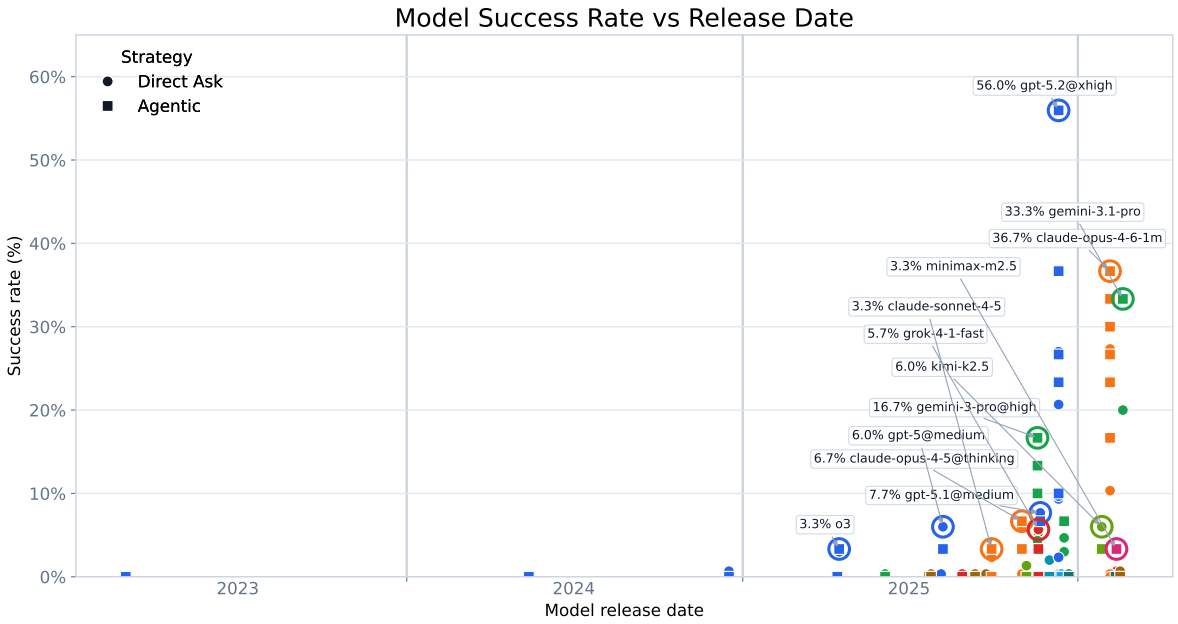


Figure 7: Full model success rates over time with frontier model release dates annotated, including all evaluated models. Models released before late 2024 show 0% or near-0% solve rates, with capability emerging only in late 2025.