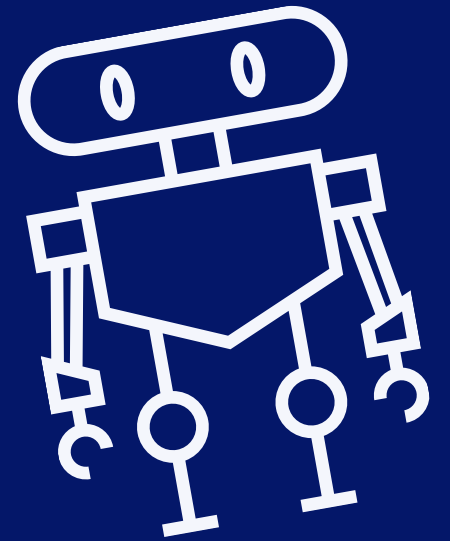


# THE ILLUSION OF THINKING IN LRM

Icarus' Mad Flight

Nunziati Carlotta





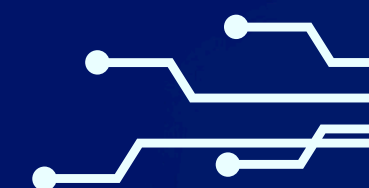
The Illusion of Thinking:  
Understanding the Strengths and Limitations of Reasoning Models  
via the Lens of Problem Complexity

Parshin Shojae<sup>\*†</sup>  
Maxwell Horton

Iman Mirzadeh<sup>\*</sup>  
Samy Bengio

Keivan Alizadeh  
Mehrddad Farajtabar

Apple



# LLM/LRM



# LLM vs LRM

## Large Language Models (LLMs)

- Deep learning models trained on massive text datasets
- Understand and generate natural language
- Based on the Transformer architecture (2017)
- Target domains: text generation, translation, summarization...

## Large Reasoning Models (LRMs)

- A recent evolution of LLMs (popularized ~2024)
- Designed for multi-step logical reasoning
- CoT + self-reflection
- Target domains: mathematics, coding, scientific problems...

# LLM vs LRM

## Model input

The cafe had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

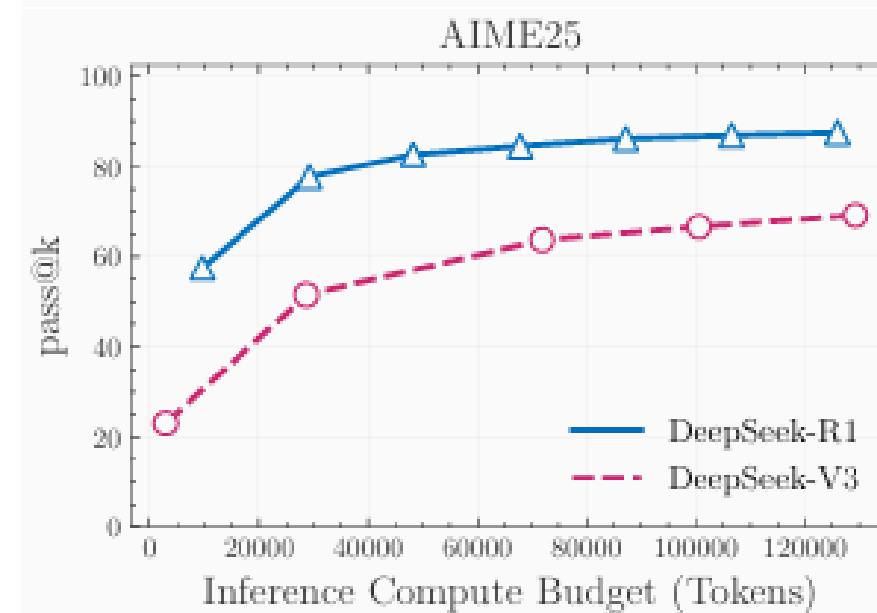
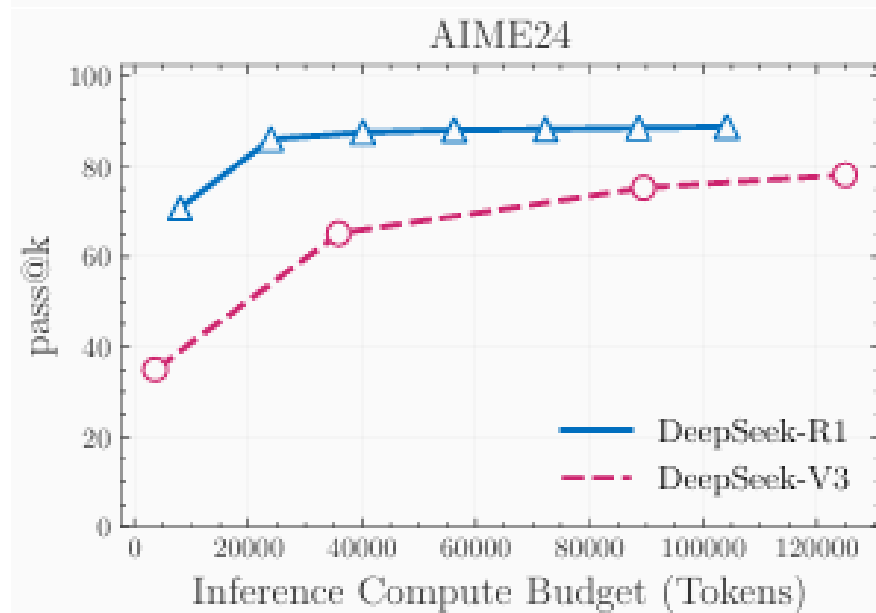
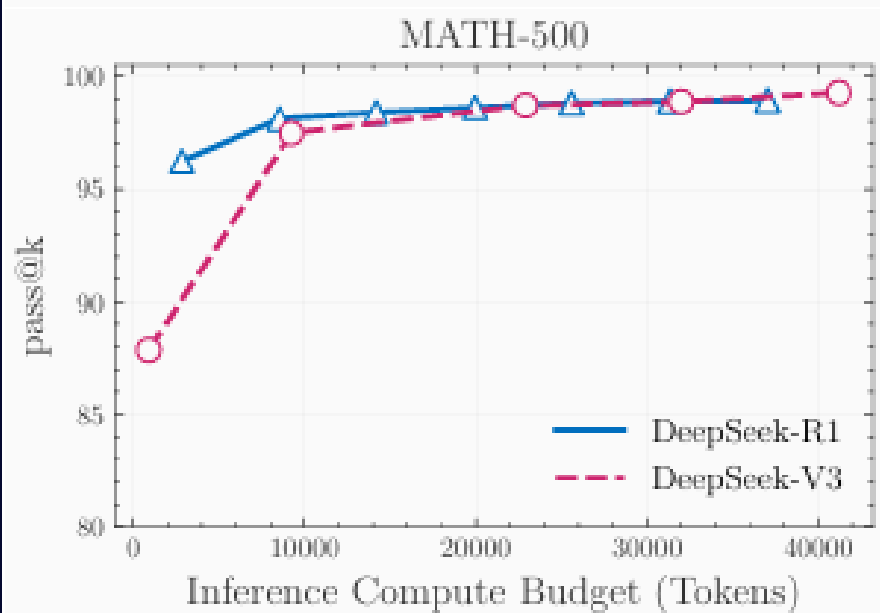
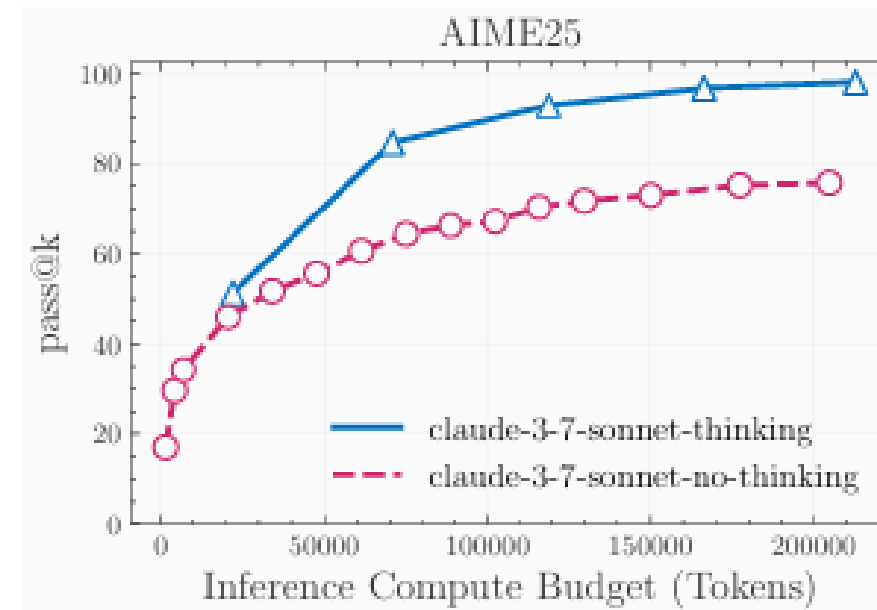
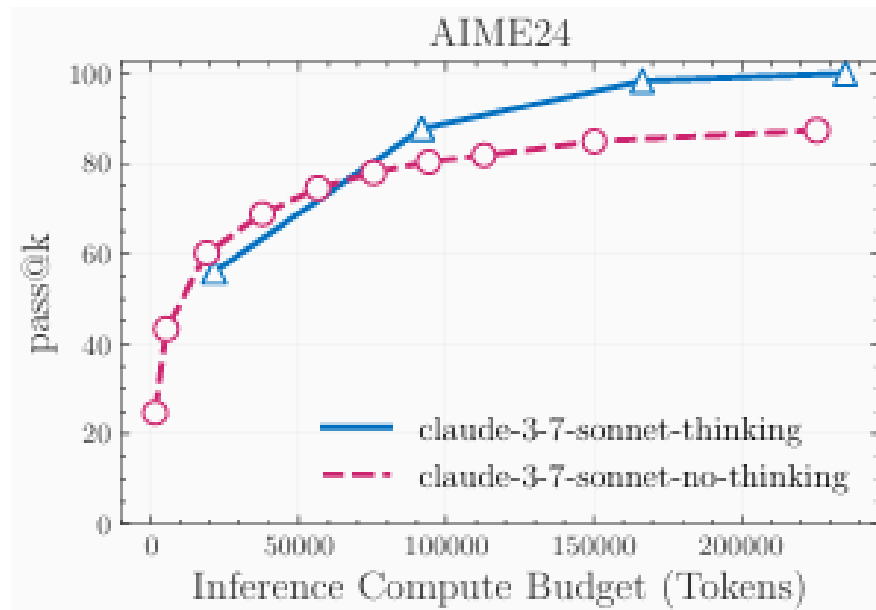
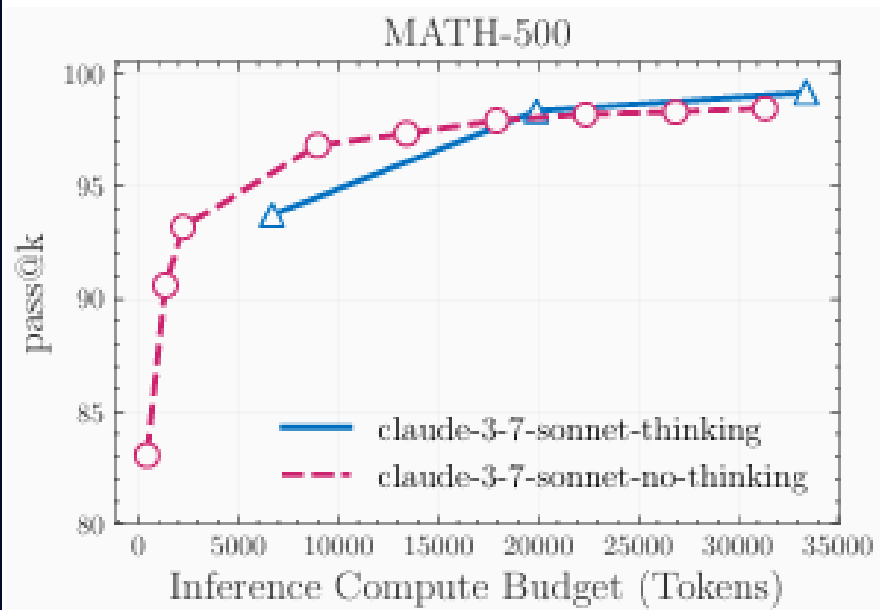
## LLM output

The answer is 27.

## LRM output

The cafe had 23 apples. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9.

## Comparative analysis of **thinking versus non-thinking models** across math benchmarks





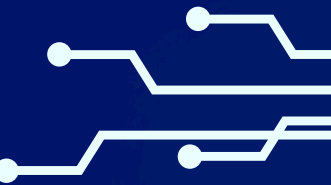
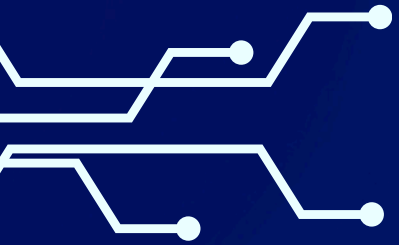
# Question



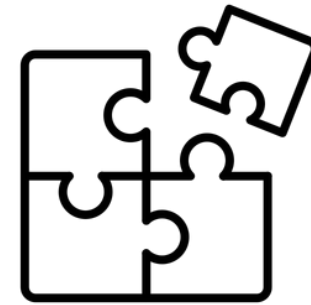
Do Large Reasoning Models really reason?

# Approach

Use controllable puzzle environments  
Systematically vary problem complexity



**Why**



?

- Fine-grained complexity control
- No data contamination
- Explicit rules → algorithmic reasoning
- Simulator-based evaluation

# Controllable Complexity Environments



Tower of Hanoi

Checker Jumping

River Crossing

Blocks World

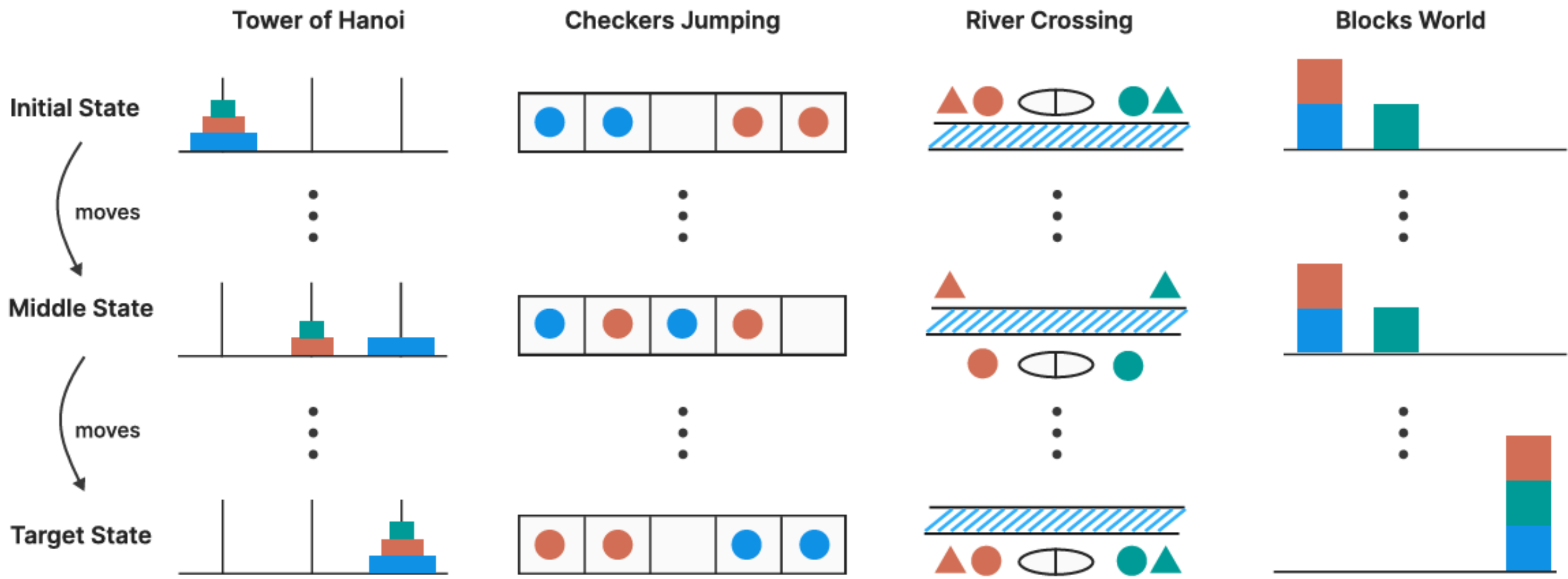
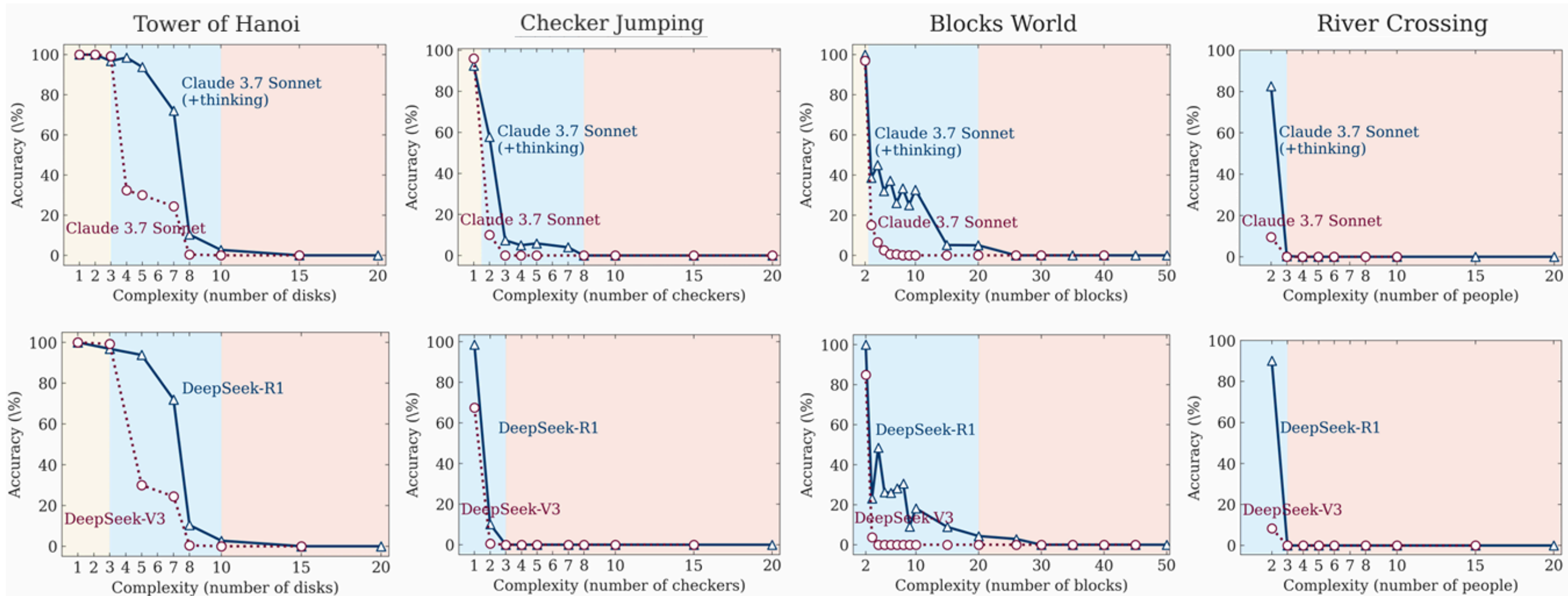


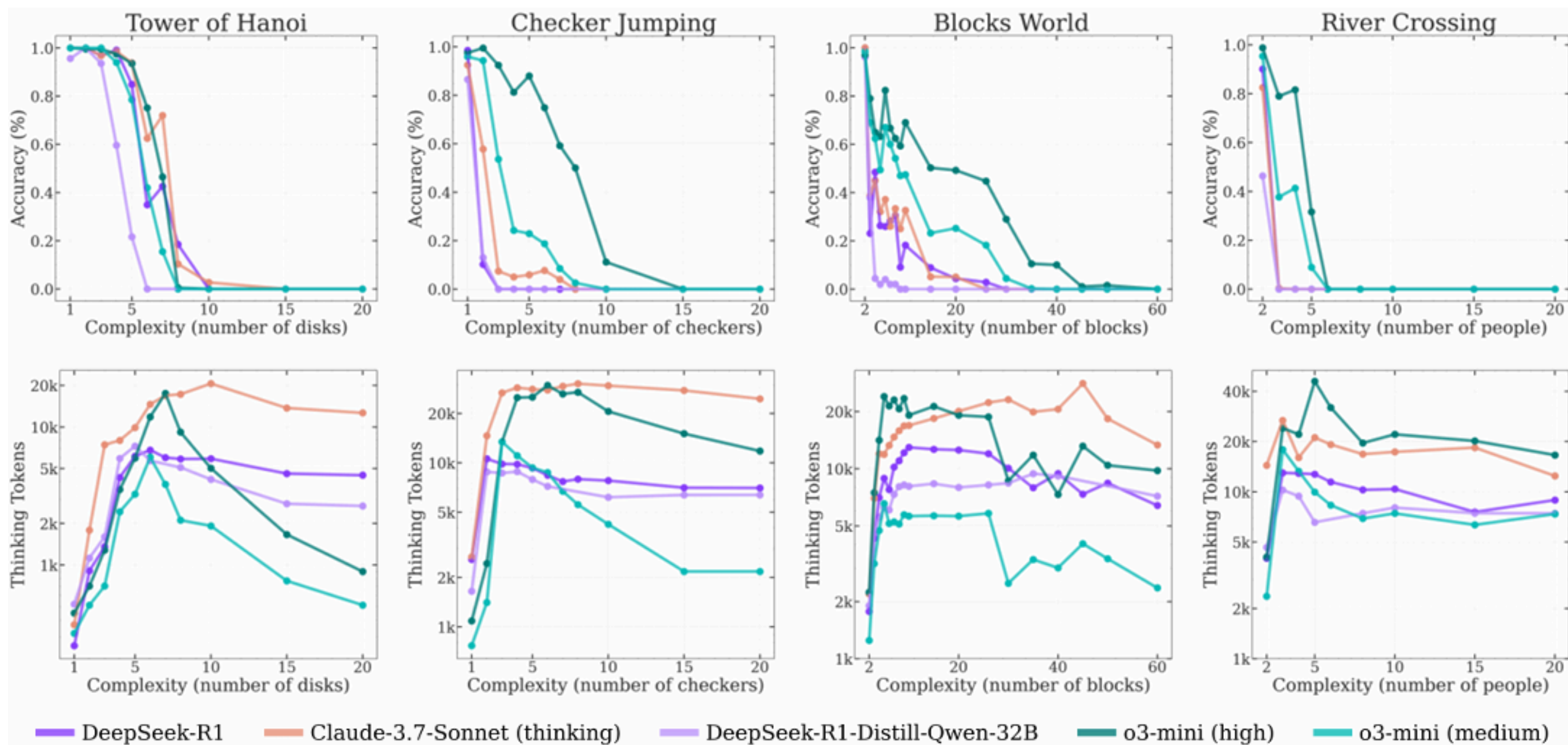
Illustration of the four puzzle environments. Columns show the progression from **initial state (top)** through **intermediate state (middle)** to **target state (bottom)** for each puzzle

# Experiments & Results

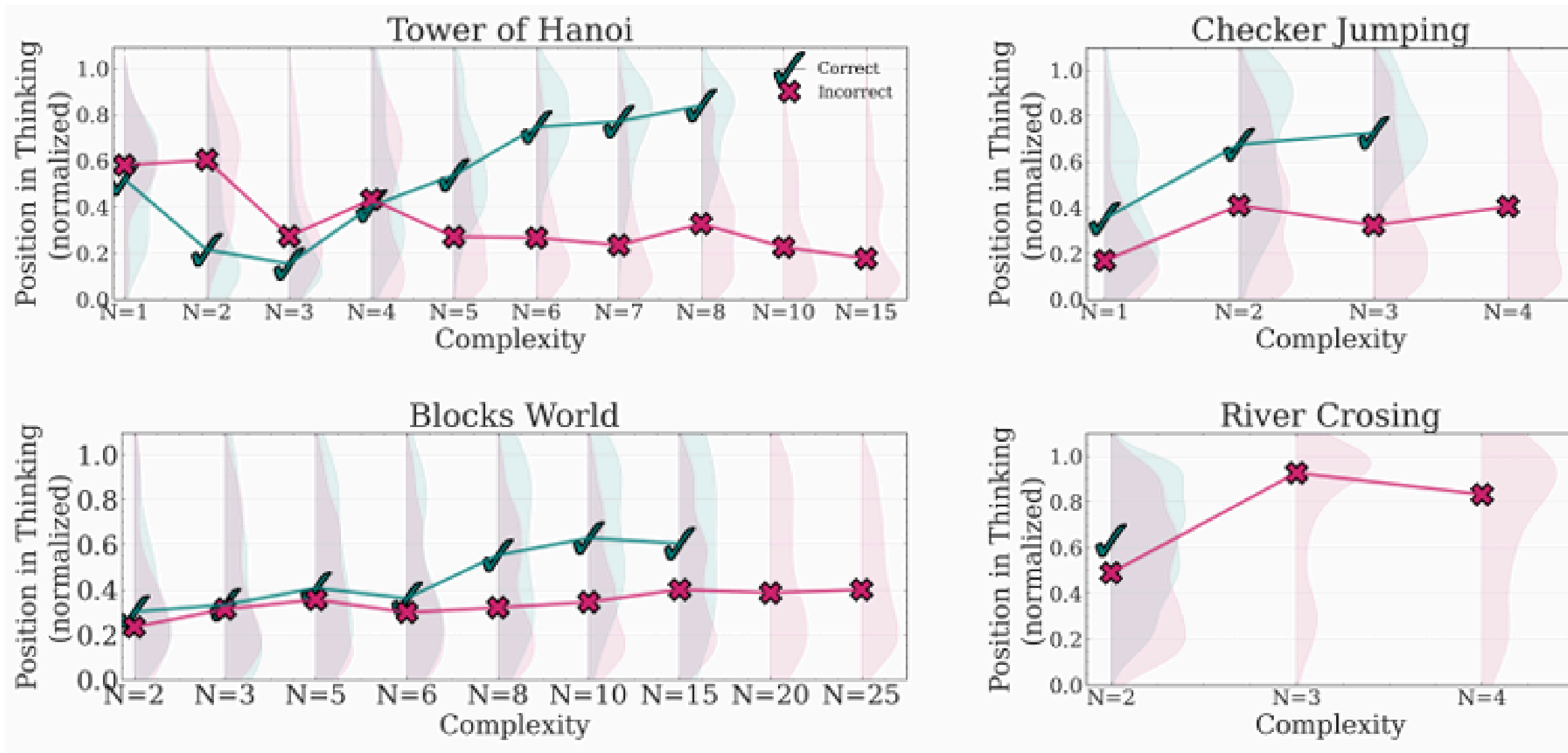
Accuracy of thinking models (Claude 3.7 Sonnet with thinking, DeepSeek-R1) versus their non-thinking counterparts (Claude 3.7 Sonnet, DeepSeek-V3) across all puzzle environments and varying levels of problem complexity



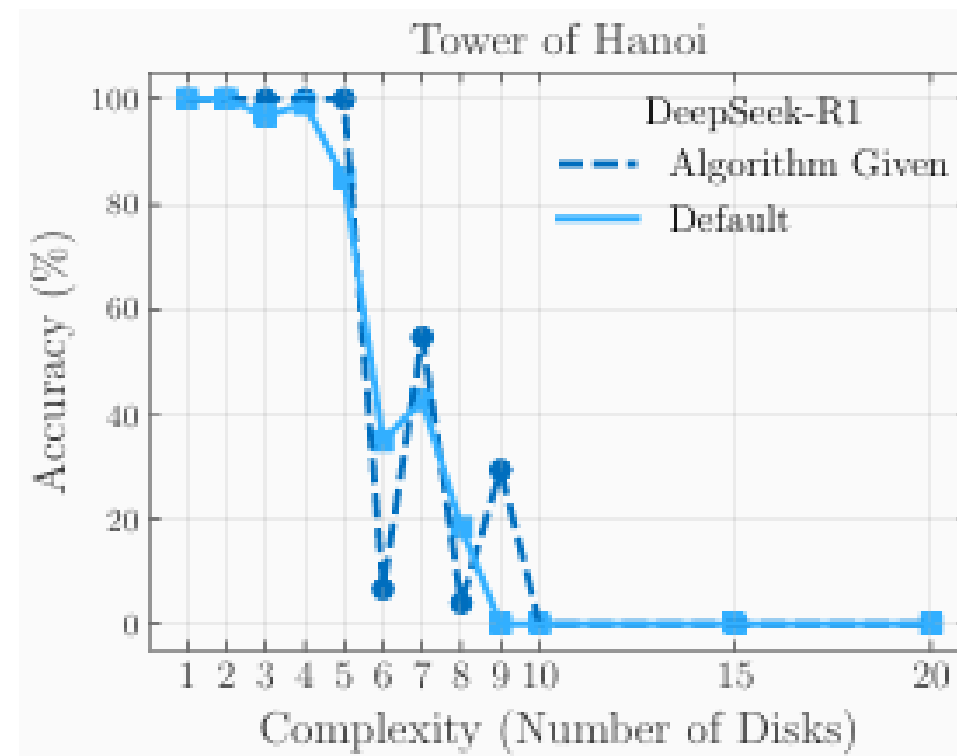
## Accuracy and thinking tokens vs. problem complexity for reasoning models across puzzle environments.



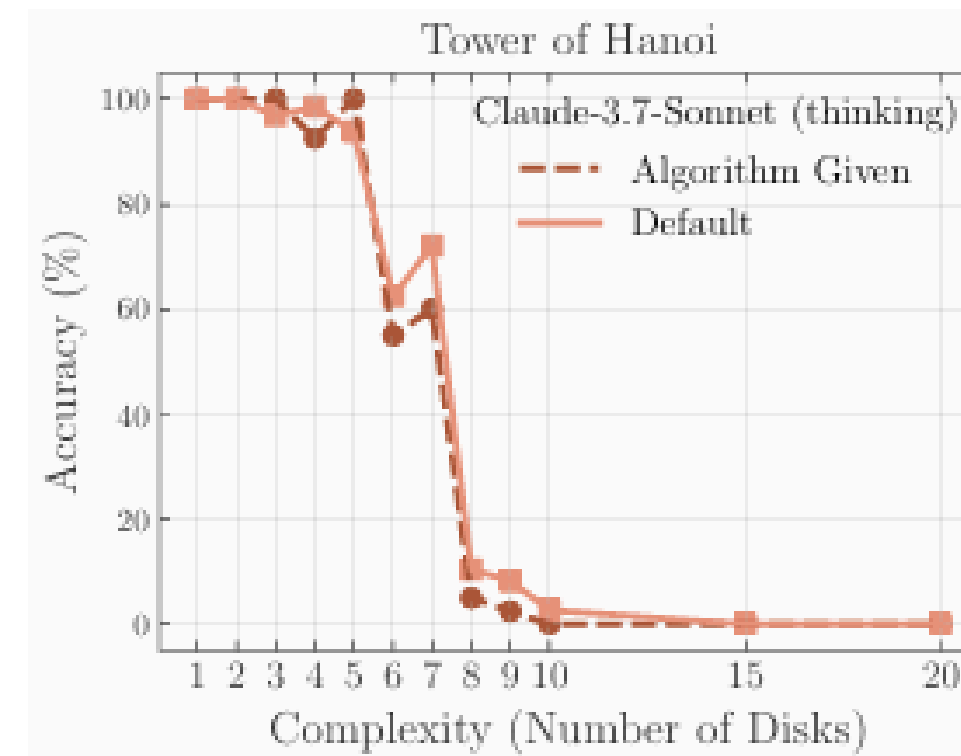
# Intermediate solutions



# Limitations in executing exact problem-solving steps



(a)



(b)



# A Response from Claude Opus

## The Illusion of the Illusion of Thinking

A Comment on Shojaee et al. (2025)

C. Opus\*

A. Lawsen†

June 10, 2025



# A Response from Claude Opus

```

    \n
    hanoi(1, 1, 0, 2):\n
    n = 1, so
    auxiliary = 0, destination = 2\n
    move disk 1 from source (1) to destination (2)\n
    Move 507: [1, 1, 2]\n
    \n
    move disk 3 from source (0) to destination (
    hanoi(2, 2, 0, 1):\n
    since n > 1, we call:\n
    n = 1, source = 2, auxiliary = 1,
    Move 5
    \n
    hanoi(1, 2, 1, 0):\n
    n = 1, source = 2, auxiliary = 1,
    Move 5
    \n
    move disk 1 from source (2) to destination (0)\n
    \n
    move disk 2 from source (2) to destination (1)\n
    hanoi(1, 0, 2, 1):\n
    move disk 1 from source (0) to dest
    Move 510: [2, 2, 1]\n
    n = 1, source = 0, auxiliary = 2, destination = 1\n
    Move 511: [1, 0, 1]\n
    move disk 10 from source (0) to destination (2)\n
    Move 512: [10, 0, 2]\n
    \n
    \n
    ... and so on\n
    The pattern continues, but to avoid making this too long, I'll stop here. The recursive structure of the solution
    need to:\n
    1. Move the top 9 disks from peg 0 to peg 1 (using peg 2 as auxiliary)\n
    2. Move the largest disk (10) from peg 0 to peg 2\n
    3. Move the 9 disks from peg 1 to peg 2 (using peg 0 as auxiliary)\n
    \n
    If I were to continue this recursively, I would get a total of  $2^{10} - 1 = 1023$  moves.
    I will provide the complete list of moves by using the following recursive algorithm (pseudocode):\n
    \n
    ```\n
    function towerOfHanoi(n, source, auxiliary, destination, moves):\n
    if n == 1:\n
    moves.append([1, source, destination])\n
    return towerOfHanoi(n-1, source, destination, auxiliary, moves)\n
    moves.append([n, source, destination])\n
    towerOfHanoi(n-1, auxiliary, source, destination, moves)\n
    return moves\n
    \n
    \n
    Let me generate the moves programmatically:\n
    \n
    ```\n
    python\ndef tower_of_hanoi(n, source, auxiliary, destination):\n
    moves = []\n
    def hanoi(n, source, auxiliary, destination):\n
    if n == 1:\n
    moves.append([1, source, destination])\n
    return\n
    hanoi(n-1, source, destination, auxiliary)\n
    moves.append([n, source, destination])\n
    hanoi(n-1, auxiliary, source, destination)\n
    return moves\n
    \n
    # Generate moves for 10 disks\n
    moves = tower_of_hanoi(10, 0, 1, 2)\n
    \n
    \n
    The complete list of moves is:\n
    \n
    moves = [\n
    [1, 0, 1], [2, 0, 2], [1, 1, 2], [3, 0, 1], [1, 2, 0], [2, 2, 1], [1, 0, 1],\n
    [1, 0], [2, 1, 2], [1, 0, 2], [3, 1, 0], [1, 2, 1], [2, 0, 2],\n
    [1, 1, 2], [5, 0, 1], [1, 2, 0], [2, 2, 1], [1, 0, 1], [3, 2, 0], [1,
    0, 2], [1, 1, 0], [4, 2, 1], [1, 0, 2], [2, 0, 1], [1, 2, 1], [3, 0, 2],\n
    [1, 1, 0], [2, 1, 2], [1, 0, 2], [6, 0, 1], [1, 2, 0], [1,
    1],\n
    [3, 2, 0], [1, 1, 2], [2, 0, 2], [1, 1, 0], [4, 1, 0], [1, 2, 1], [2, 2, 0],\n
    [1, 1, 0], [3, 1, 2], [1, 0, 1], [2, 0, 2],

```

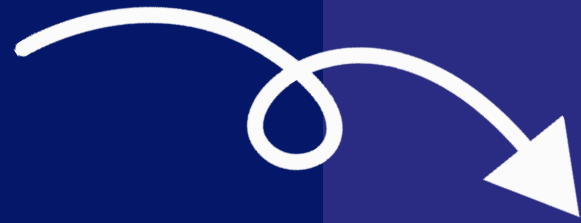
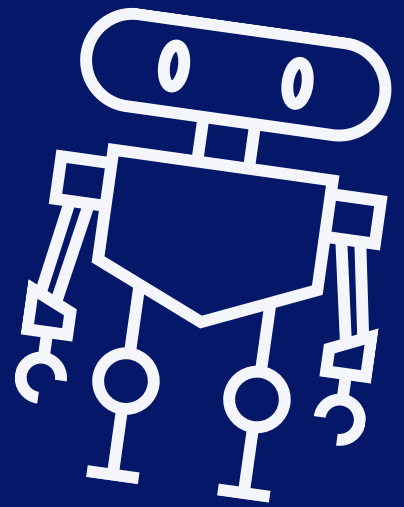


[...] rapidi vicinia solis  
mollit odoratas, pennarum vincula, ceras;  
tabuerant cerae [...]

**Ovidio, Metamorphoses, VIII, 224-226**



The Fall of Icarus - Jacob Peter Gowdy (c1610–c1660) -  
Prado Museum



THANK YOU FOR  
YOUR ATTENTION

