

AAAI 2025 Bridge on Constraint Programming and Machine Learning

A Neurosymbolic Fast and Slow Architecture for Graph Coloring

Vedant Khandelwal, Vishal Pallagani, Biplav Srivastava, Francesca Rossi



UNIVERSITY OF
SOUTH CAROLINA

IBM Research



Paper

Outline

- Overview
- SOFAI Architecture Evolution
- Experimental Setup
- Results
- Future Work

- System 1 can perform cognitively easy tasks
 - Local and parallel
 - Handles and exploits causality to build approximate models of the world
- System 2 handles more complex tasks
 - Requires all our attention
 - Global and sequential
- System 2's search for solutions is usually supported by System 1's heuristics
- Skill learning: some tasks transfer from System 2 to System 1 over time
 - Not all (ex.: complex arithmetic operations)
- Cognitive control
 - System 1 reacts, System 2 may override it

The Role

To monitor and control:

- cognitive activities
- processes
- structures

Main Goal

To improve the System's decisions quality

Our Choice

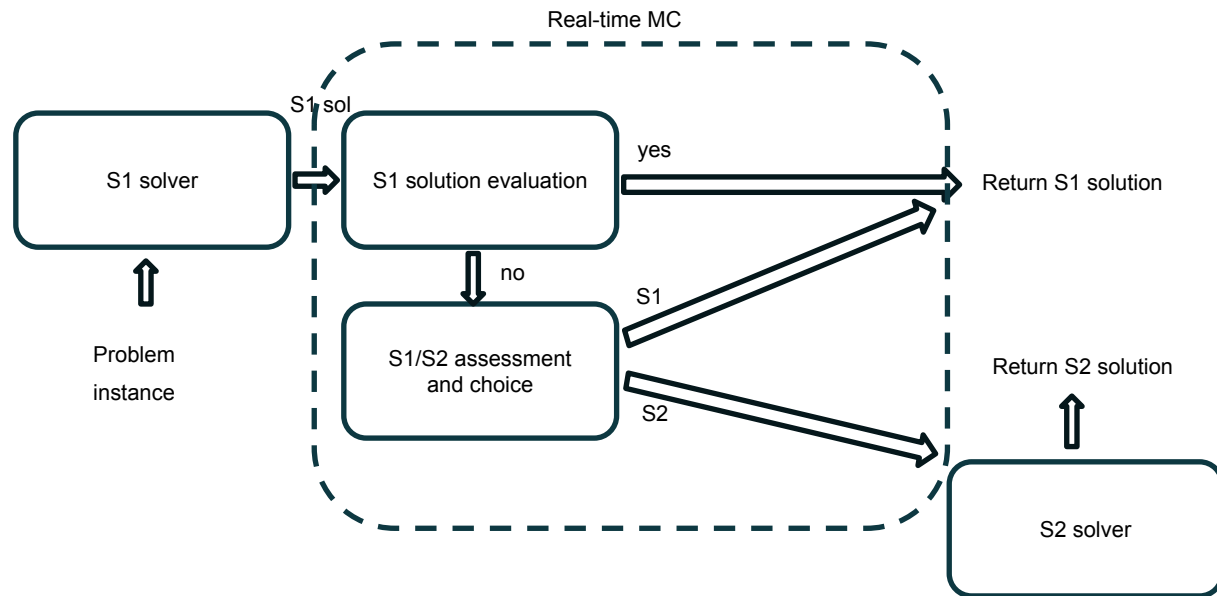
- Centralized meta-cognitive agent
- Exploiting both internal and external information
- Arbitrates between System 1 and System 2 solvers

What is SOFAI? A Neurosymbolic Approach

- SOFAI (Slow and Fast AI) is a neurosymbolic architecture combining learning-based and symbolic solvers.
- Inspired by Kahneman's Thinking Fast & Slow, it integrates System 1 (fast, heuristic-based) and System 2 (slow, deliberative reasoning) for solving CSPs.
- Why SOFAI for Graph Coloring?
 - Traditional methods are either too slow (symbolic) or inaccurate (neural).
 - SOFAI dynamically decides when to use fast neural inference or slow symbolic search.

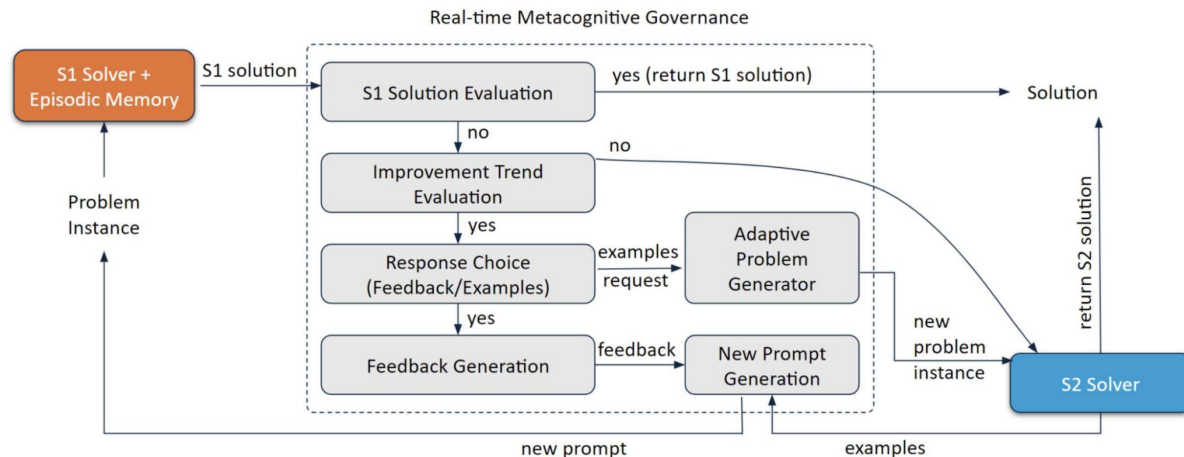
SOFAI Evolution

- New problem instance goes to S1 solver, which provides a candidate solution.
- MC evaluates the S1's candidate solution: if not acceptable, calls an S2 solver



Generalizable SOFAI Architecture (SOFAI v2)

- Added Episodic memory, which holds past experience.
- S1 solver is used more than once by MC:
 - Mistakes in the candidate solution are provided as feedbacks.
 - Simpler examples are provided using S2.
- On exhausting the limit on number of calls for S1 or no improvement over past few calls, S2 is involved.



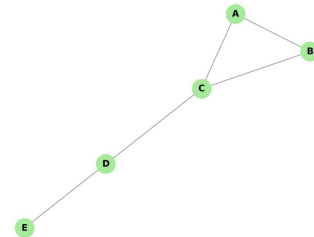
Experimental Setup

Graph Coloring Problem

- Objective: Given a graph and number of colors (n), find a coloring of the nodes (if exists) such that no two adjacent nodes are colored with the same color using at most n colors.
- Graphs are generated using the Erdős–Rényi model to generate graphs with nodes (5-50), and with edge probability ($[0.1-0.9]$) in DIMACS format.
- Chromatic Number (n): minimum number of colors required to solve the problem.
- Chromatic Number is computed using DSATUR Algorithm
- Solvability mix (m):
 - (100%, 0%) set: all graphs where the allowed colors $\geq n$ (fully solvable).
 - (0%, 100%) set: all graphs where allowed colors $< n$ (unsolvable).
 - (50%, 50%) set: half are solvable, half unsolvable.

```
1 c Example of a graph coloring
2   problem in DIMACS format
3 p edge 5 5
4 c edges
5 e A B
6 e A C
7 e B C
8 e C D
9 e D E
```

(a) DIMACS Format of Graph Coloring Problem



(b) Visualization of the Graph

- **Baseline solvers:**
 - System 1 (S1) : LLM (Mistral-7B) to quickly propose a color assignment.
 - System 2 (S2) : Degree of Saturation (DSATUR) with backtracking.
 - Finds a valid assignment or proves none exists.
 - Guarantees correctness by systematically exploring color choices (hence “slow”).
 - SOFAI-v1: Initial SOFAI architecture, which invokes S2 if the candidate solution from S1 is incorrect.
- **SOFAI-v2: Iteratively refines S1 attempts (via feedback and examples).**
 - S1 solver: LLM (Mistral - 7B) with episodic memory.
 - S2 solver: DSATUR with backtracking.
 - If repeated tries fail, calls S2 as a fallback.
- **Each solver has a time limit of 200 seconds to solve any given problem.**

Graph Coloring Problem Prompt with Episodic Memory Example

New Problem to Solve:

You are given an undirected graph with 2 colors available. Your task is to assign a color to each vertex such that no two adjacent vertices share the same color.

Graph Representation:

- Number of vertices and edges: p edge 5 5.
- Edges between vertices are listed as follows:
e A B
e A C
e B C
e C D
e D E

Objective:

Assign a unique color to each vertex, ensuring that no two vertices connected by an edge have the same color. Use no more than 2 distinct colors. Provide the color assignments for each vertex in the format:
(Vertex Color)

Episodic Memory Example:

Graph: p edge 3 3

Edges:

- e A B
 - e B C
 - e C A
- Solution:**
(A 1)
(B 2)
(C 3)

This example is provided to demonstrate a previously successful coloring strategy for a similar subgraph, which may be helpful in solving the current problem.

Please provide the color assignment for the new problem to solve directly below, or respond with "NOT SOLVABLE" if it cannot be solved.

Example Input

Example Targeted Feedback & Examples

Graph Coloring Feedback – Incorrect Coloring

Given an undirected graph with 2 colors available, the vertices were incorrectly colored as follows:

Graph Representation:

- Number of vertices and edges: p edge 5 5.
- Edges between vertices are listed as follows:
e A B
e A C
e B C
e C D
e D E

Incorrect Coloring Submitted:

- (A 1)
- (B 1)
- (C 2)
- (D 2)
- (E 1)

Feedback Provided:

Error: Vertices A and B are adjacent but have the same color.
Error: Vertices B and C are adjacent but have the same color.

Graph Coloring Generated Example

To generate an example, a subgraph is extracted from the problem graph to demonstrate a valid coloring with fewer vertices.

Subgraph Representation:

- Subgraph vertices and edges: p edge 3 3.
- Edges between subgraph vertices are listed as follows:
e A B
e A C
e B C

Correct Coloring of Subgraph:

- (A 1)
- (B 2)
- (C 3)

This example shows how to correctly assign colors in a smaller scope of the main problem. Scaling this approach for larger graphs while maintaining the constraints is crucial.

Graph Coloring Feedback - Exceeding Limit

An attempt was made to solve the graph coloring using three colors, exceeding the maximum allowed.

Over-Colored Submission:

- (A 1)
- (B 2)
- (C 3)
- (D 1)
- (E 2)

Feedback Provided:

Error: Only 2 colors are allowed. 3 colors were used.

Results

RQ1: Improvement in Success Rate

Graph Size	m(100,0) (in %)				m(0,100) (in %)				m(50,50) (in %)			
	S1	S2	SOFAL_v1	SOFAL-v2	S1	S2	SOFAL_v1	SOFAL-v2	S1	S2	SOFAL_v1	SOFAL-v2
5	9.41	100	100	100	75	100	100	100	45.36	100	100	100
10	0	100	100	100	60.38	100	100	100	39.39	100	100	100
15	0	80	80	80	37.50	77.08	89.58	93.75	17.65	77.45	83.33	84.31
20	0	5	5	7.29	45.65	4.35	47.83	76.09	22.58	3.06	24.73	38.95
30	0	0	0	0	54.17	0	62.50	72.92	28.89	0	33.71	38.89
40	0	0	0	0	33.33	0	47.37	47.37	17.43	0	24.55	24.77
50	0	0	0	0	3.85	0	3.85	53.85	2.11	0	2.11	27.72

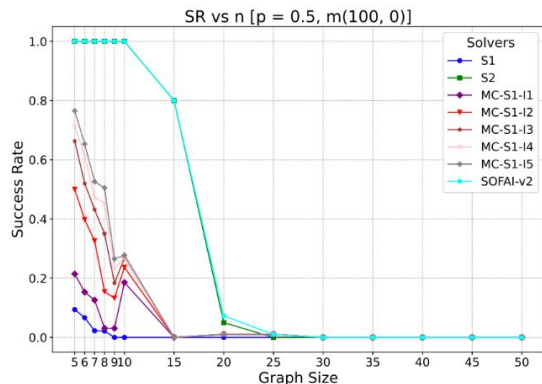
- SOFAI-v2 significantly outperforms other solvers in mixed and unsolvable cases, achieving up to 53.85% success rate in large unsolvable graphs where others fail.
- For smaller graphs, all solvers perform well, but SOFAI-v2 shows improvement in handling problems of higher complexity.

RQ2: Improvement in Time Efficiency

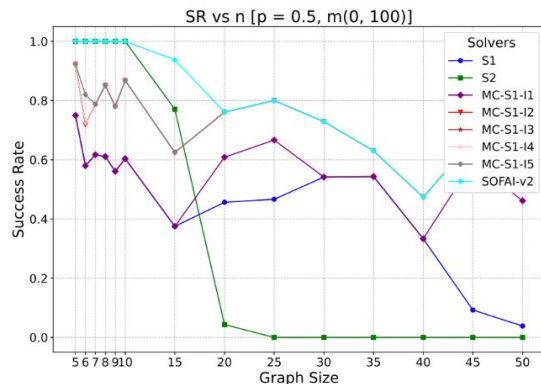
Graph Size	m(100,0) (in seconds)				m(0,100) (in seconds)				m(50,50) (in seconds)			
	S1	S2	SOFAL_v1	SOFAL-v2	S1	S2	SOFAL_v1	SOFAL-v2	S1	S2	SOFAL_v1	SOFAL-v2
5	3.23	0.001	3.23	0.92	3.13	0.001	3.13	0.77	3.23	0.001	3.23	0.92
10	4.56	0.19	4.69	2.21	4.08	0.17	4.14	1.07	5.28	0.22	5.48	3.74
15	5.61	70.85	55.46	53.23	6.20	72.02	39.89	22.33	5.12	67.08	70.01	85.37
20	6.79	199.27	154.11	120.86	7.52	198.70	111.00	46.96	6.68	196.22	193.15	195.25
30	11.53	200.00	137.80	117.10	10.87	200.00	82.30	52.10	9.92	200.00	192.85	193.67
40	9.69	200.00	153.41	144.54	9.46	200.00	109.71	101.18	10.30	200.00	194.47	193.67
50	41.98	200.00	200.00	139.17	62.76	200.00	88.93	88.93	21.83	200.00	200.00	194.47

- SOFAI-v2 reduces time complexity compared to pure symbolic solvers, especially for unsolvable cases where S2 alone reaches its 200s timeout.
- For smaller graphs, all solvers perform efficiently, but as graph size increases, SOFAI-v2 remains significantly faster than S2 and SOFAI-v1 while maintaining correctness.
- SOFAI-v2 efficiently balances speed and accuracy, leveraging S1's speed while ensuring constraint adherence through iterative refinement before invoking S2.

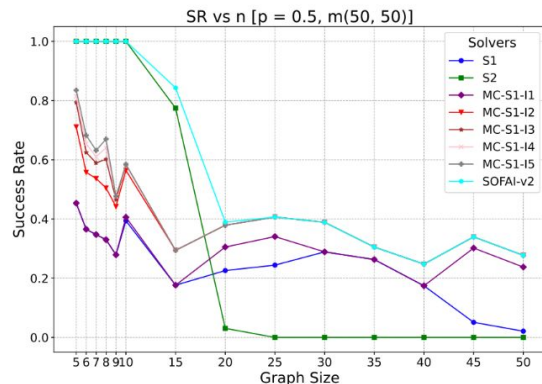
RQ2: Improvement with Iterative calling



(a) All Solvable ($m = (100, 0)$)



(b) All Unsolvable ($m = (0, 100)$)



(c) Balanced Mix ($m = (50, 50)$)

- Iterative calling via metacognitive governance improves S1's success rate across all problem configurations, demonstrating that guided refinement leads to more correct solutions.
- As the number of metacognitive iterations (MC-S1-I1 to MC-S1-I5) increases, S1's performance improves, especially in unsolvable and mixed cases, reducing reliance on S2.
- SOFAI-v2 consistently outperforms standalone S1 and earlier feedback mechanisms, confirming that metacognitive governance enhances problem-solving efficiency and accuracy.

- Extend SOFAI-v2 to other CSPs like SAT solving and planning to assess its generalizability.
- Optimize the metacognitive module with reinforcement learning to enhance adaptive feedback control.
- Using existing knowledge and rules to better generalize to other problems.

Thank You

Questions?

Contact Information

Vedant Khandelwal –
vedant@email.sc.edu



Paper