Comparing Rubik's Cube Solvability in Domain-Independent Planners Using Standard Planning Representations for Insights and Synergy with Upcoming Learning Methods

Bharath Muppasani, Vishal Pallagani, Biplav Srivastava, Forest Agostinelli

AI Institute, University of South Carolina, USA

ICAPS 2024

Banff, Canada

Outline

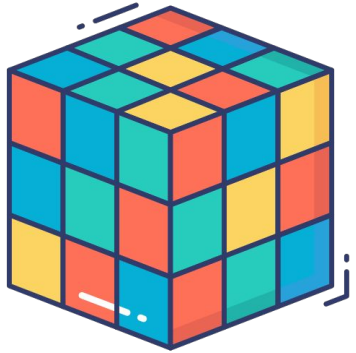
- Introduction / Motivation
- Background
- Rubik's Cube
  - Representations
  - Heuristics Considered
  - Results Analysis
- Beyond RC - Sokoban
- Discussion and Conclusion

01.

Introduction / Motivation

Introduction / Motivation

- Combinatorial puzzles, like Sudoku, the Rubik's Cube, and N-Puzzle, often have vast numbers of possible configurations, posing significant computational challenges.
- The **Rubik's Cube**, with over **43 quintillion** possible states, serves as a prime example of this complexity.
- Navigating the immense state space efficiently poses significant computation challenges, especially for time and memory constraints.



- Achieving an optimal solution often conflicts with computational feasibility, highlighting the need for trade-offs between solution optimality and time taken.
- In our work, we **compare** different existing **problem representations for the Rubik's Cube** and solve the IPC-2023 Rubik's Cube problem dataset **using various compatible heuristics**. Our goal is to determine the best representation and heuristic combination for providing optimal solutions. Additionally, we also compare against the best learning-based solver, **DeepCubeA**.
- To assess the general applicability of our insights, we started exploring beyond the Rubik's Cube, focusing on the **Sokoban puzzle**.

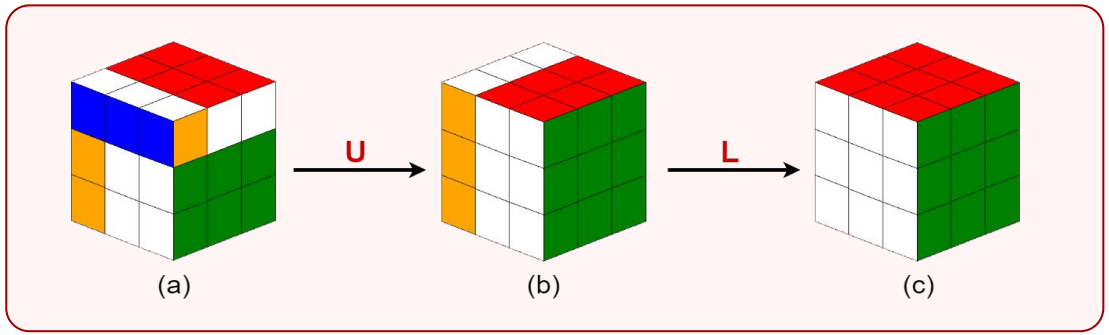
02.

Background

Background

Rubik's Cube: is a 3-D puzzle made up of 26 smaller colored pieces anchored to a central spindle.

- The primary actions are Up (U), Down (D), Right (R), Left (L), Front (F), and Back (B), each representing a 90-degree clockwise turn, with their inverses indicating a counter-clockwise rotation.
- Starting from a scrambled state, the goal is to perform a series of these rotations to return the cube to its solved state, where each face is a single color.

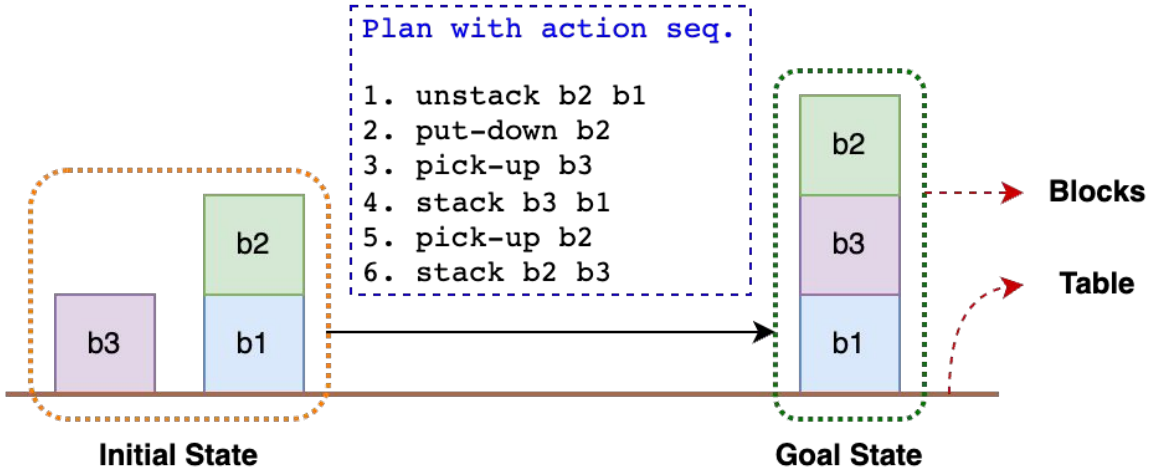


Shuffled state of the cube. Solution found with FD planner: U, L. Cost: 2.

Background

Automated Planning: also known as AI planning, is the process of finding a sequence of actions that will transform an initial state of the world into a desired goal state, given the world model.

Key elements: Goals, Actions, States, Constraints



Background

Heuristic Search: uses a heuristic function to estimate the cost to reach the goal state from a given state. It helps in choosing actions that are likely to lead to a goal more efficiently.

Admissible Heuristics: never overestimate the cost to reach the goal, ensuring optimal solutions.

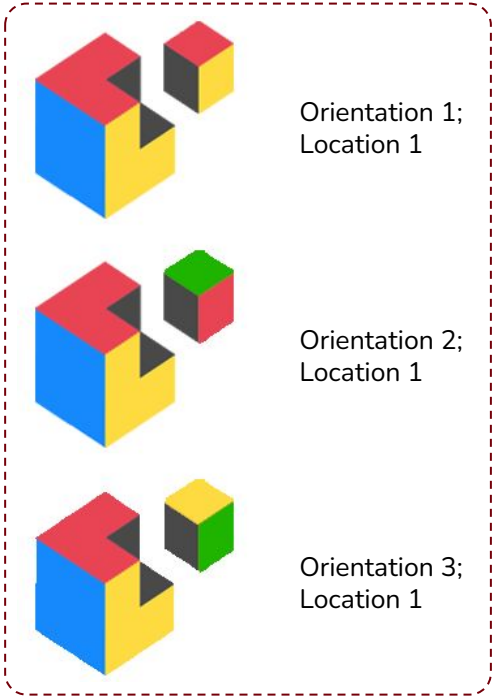
Inadmissible Heuristics: overestimate the cost, potentially finding solutions faster but not guaranteeing optimality.

03.

Rubik’s Cube Representations

SAS Model*

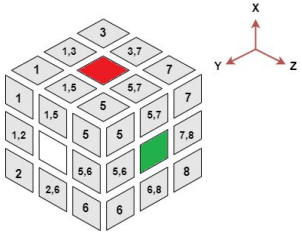
- SAS+ representation as a factored effect task.
- Action modeling
  - Considered the change in orientation and location of cube pieces as action effect.
- Each cube piece is considered as a variable, and different values are assigned based on orientation and location of the cube piece.
  - Corner pieces - 3 orientations, 8 locations - 24 values
  - Edge pieces - 2 orientations, 12 locations - 24 values
- State representation using - 20 Variables (8 Corner pieces, 12 Edge pieces); 480 Fact Pairs - **16 Bytes memory.**

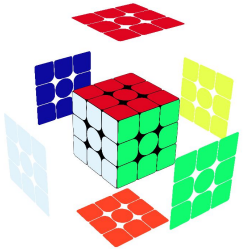


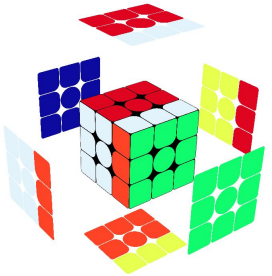
*Büchner, C., Ferber, P., Seipp, J. and Helmert, M., 2022, April. A Comparison of Abstraction Heuristics for Rubik's Cube. In *ICAPS 2022 Workshop on Heuristics and Search for Domain-independent Planning*.

PDDL Model

- Used conditional effects in PDDL modeling.
- Action Modeling
  - Fixed cube piece positions
  - Colors as objects - change of colors on cube pieces as effect of an action.
- State representation after translating to SAS by FastDownward Planner
  - 480 variable (binary) - 960 fact pairs
  - **60 bytes memory**

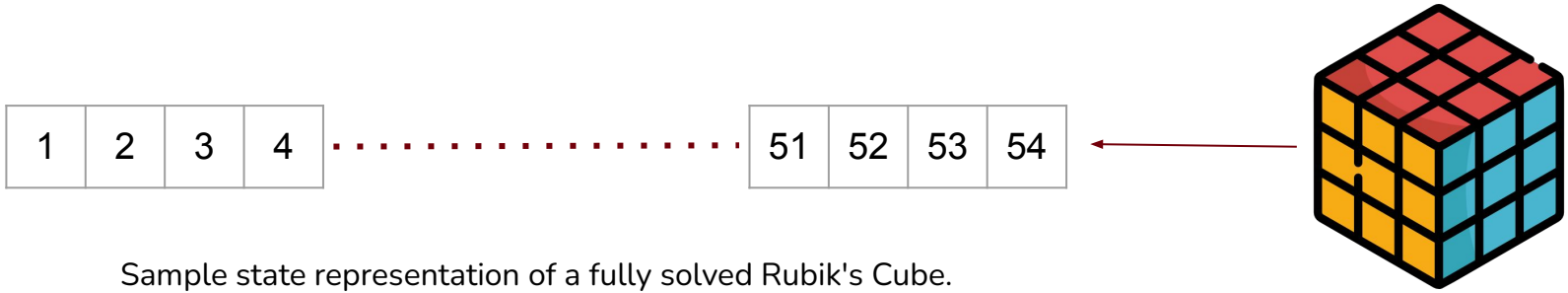






DeepCubeA representation

- Considered a one-hot encoding for the state representation
- All the 54 color values of the Rubik's Cube are represented in a unidimensional array.



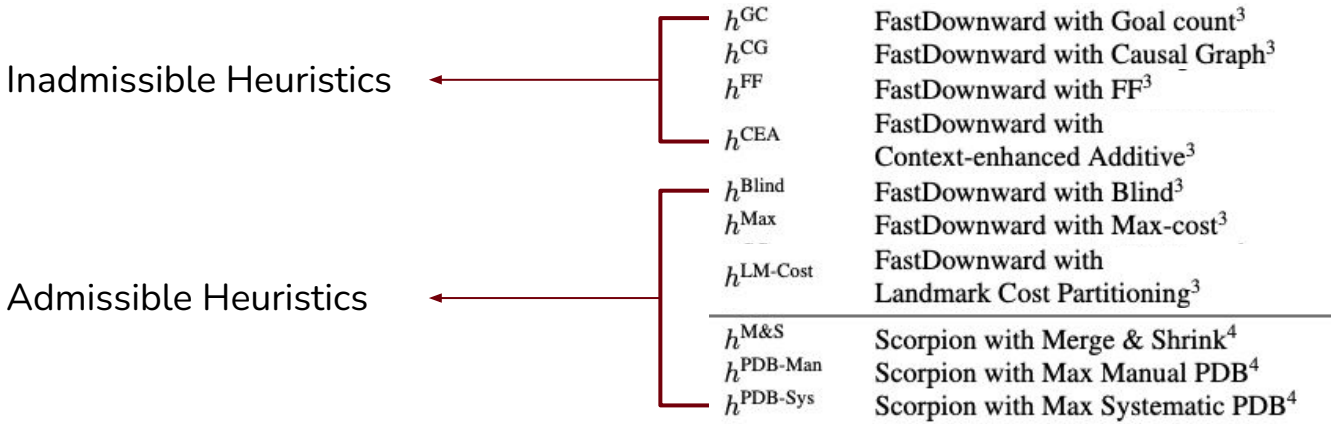
Sample state representation of a fully solved Rubik's Cube.

04.

Heuristics Considered for Evaluation

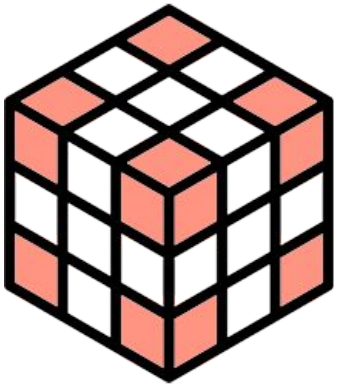
Heuristics

- We considered evaluating various heuristics available in the FastDownward planning system that are compatible with conditional effects in domain modeling.

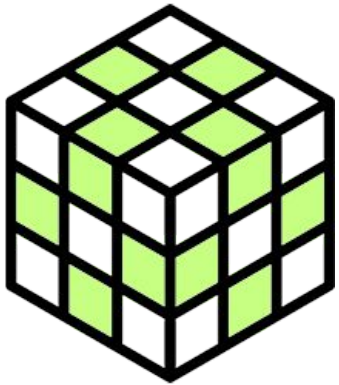


Pattern Database Heuristic

- Two version -
  - $h^{\text{PDB-MAN}}$ manual patterns - Inspired by *Korf's patterns** for Corner and Edge pieces.
  - $h^{\text{PDB-SYS}}$ automatically generates interesting patterns



Corner pieces



Edge pieces

Korf, R.E., 1997, July. Finding optimal solutions to Rubik's Cube using pattern databases. In AAAI/IAAI (pp. 700-705).

05.

Result Analysis

Result Analysis

- **Experimental Setup**:
  - We evaluated two different Rubik's Cube representations, using various heuristics, on the IPC-2023 Rubik's Cube Dataset. For each heuristic, we performed A* searches utilizing PDDL and SAS+ domain models.
    - Time Limit / problem - 30 minutes; Memory limit / problem - 8GB.
- Abstraction Heuristics are sensitive to problem representation - Poorer in PDDL compared to SAS+
- $h^{M\&S}$ solved the same number of problems using both PDDL and SAS+, but it had a higher memory footprint for PDDL than SAS+ due to the lower memory requirement of SAS+ (16 bytes) compared to PDDL (60 bytes).
- Implications with **PDB** Heuristic:
  - *Pattern Size Limitation*: Using the SAS+ model, effective patterns can be selected with a small pattern size (1 variable per cube piece), unlike PDDL (24 variables per cube piece).
  - *Projection Complexity*: In SAS+, small pattern sizes can lead to many states due to variable multiplicity (e.g., *4* variables for corner cubies result in *331,776* states). In contrast, PDDL's binary variables require *96* variables for the same, making it inefficient and leading to increased complexity and memory usage.
- DeepCubeA solved all the tasks optimally, highlighting a key trade-off: while standard heuristics are broadly applicable to various domains with minimal computational overhead, DeepCubeA algorithm which learns domain-specific heuristics in a largely domain-independent fashion, does not generalize to other domains without the costly overhead of additional training.

Result Analysis

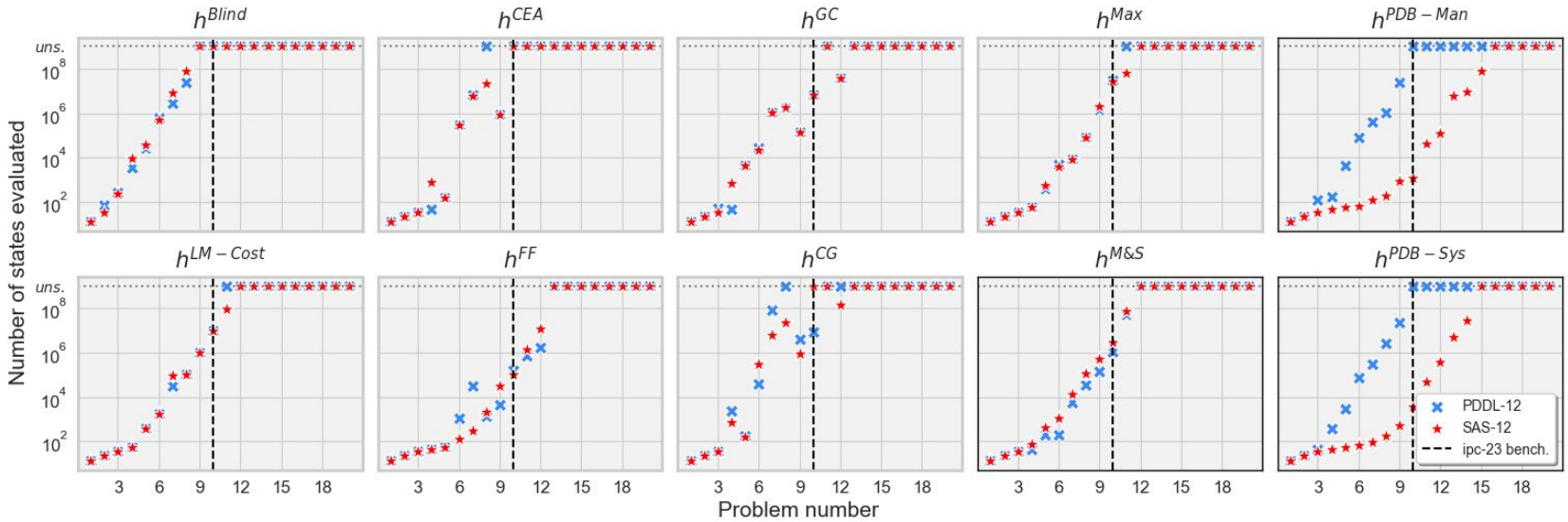
Planner with Heuristic	PDDL					SAS+				
	# Solved	Cost	# Nodes	Secs	Mem (MB)	# Solved	Cost	# Nodes	Secs	Mem (MB)
$^{+}h^{Blind}$	8 (100%)	4.5	4.52E+06	22.92	338.58	8 (100%)	4.5	1.37E+07	49.26	572.81
h^{CEA}	8 (75%)	7.38	1.08E+06	80.80	105.48	9 (55.55%)	10.11	3.84E+06	72.27	186.04
h^{CG}	9 (77.77%)	7.89	1.13E+07	165.50	1103.55	10 (50%)	11.7	1.87E+07	131.01	882.00
$^{+}h^{Max}$	10 (100%)	5.5	3.95E+06	155.12	324.82	11 (100%)	6.0	1.04E+07	136.51	543.48
$^{+}h^{LM-Cost}$	10 (100%)	5.5	1.33E+06	9.42	124.77	11 (100%)	6.0	1.13E+07	74.03	528.25
h^{GC}	11 (100%)	6.09	4.71E+06	25.84	405.09	11 (100%)	6.09	4.71E+06	11.61	245.69
h^{FF}	**12** (83.33%)	7.17	2.45E+05	11.09	38.44	12 (100%)	6.5	1.25E+06	24.46	72.40
$^{+}h^{M\&S}$	11 (100%)	6.0	6.18E+06	66.39	512.81	11 (100%)	6.0	8.19E+06	18.03	392.18
$^{+}h^{PDB-Man}$	9 (100%)	5.0	3.81E+06	20.76	278.12	**15** (100%)	8.0	7.57E+06	20.69	518.24
$^{+}h^{PDB-Sys}$	9 (100%)	5.0	3.43E+06	127.24	270.31	14 (100%)	7.5	2.88E+06	80.46	2366.62
Ragnarok*	10 (100%)	-	-	-	-	-	-	-	-	-

	# Solved	Cost	# Nodes	Secs	Mem (MB)
DeepCubeA	20 (100%)	10.5	9.67+E05	18.13	-

Comparison of the planner configurations including the total number of problems solved, along with the percentage of optimal plans, average number of nodes generated, average search time in seconds, average memory usage in MB, and average plan cost for the solved instances across different Rubik's Cube models evaluated on the IPC-2023 dataset. (* - IPC-2023 Classical optimal-track winner; + - Admissible heuristics)

State Expansion Comparison

- It can be observed in h^{PDB} the number of states expanded to find a solution is lesser with SAS+ representation than the PDDL representation.



State expansion comparison for the IPC-2023 dataset. The x-axis denotes problem numbers, and the y-axis shows the states evaluated. A top horizontal dotted line indicates unsolved problems, while a vertical dashed line marks the problems solved by the IPC-2023 Classical optimal track winner.

06.

Beyond Rubik’s Cube

Sokoban – PDDL vs DeepCubeA

- To assess the generalizability of our results, we examined the Sokoban puzzle domain from IPC-2008[1], a strategic game requiring players to push boxes to designated locations within a confined space, using heuristic based planners and DeepCubeA algorithm.
- Despite DeepCubeA's efficiency in solving the Rubik's Cube, its performance in the Sokoban domain revealed **limitations in domain adaptability**, with a success rate of only 8 out of 30 puzzles.
- This underscores the challenges in generalizing learning-based approaches across diverse problem types and highlights the **need for retraining models** for each new domain.

Planner with Heuristic	PDDL
h^{Blind}	24/30
h^{CEA}	23/30
h^{CG}	30/30
h^{Max}	28/30
$h^{LM-Cost}$	28/30
h^{GC}	27/30
h^{FF}	30/30
$h^{M\&S}$	30/30
$h^{PDB-Sys}$	26/30
DeepCubeA	8/30*

Comparative analysis of PDDL and DeepCubeA on the Sokoban domain using IPC-2008 dataset. (*noting that only 8 problems had a grid size close to 10x10, while the remainder exceeded DeepCubeA's grid size limit)

[1]https://github.com/potassco/pddl-instances/tree/master/ipc-2008/domains/sokoban-sequential-optimal-strips

07.

Conclusion

Conclusion

- **Efficiency Comparison:** Our study found that the SAS+ representation is approximately 75% more memory efficient than PDDL for solving the 3x3x3 Rubik's Cube, making it the preferred choice among standard representations.
- **Performance of Methods:** The learning-based DeepCubeA approach solved all problems optimally using its default 12-action set, while the best planner configuration achieved only 75% problem-solving with 100% optimality. However, its performance in the Sokoban domain revealed limitations in domain adaptability and generalization.
- **Future Directions:** Combining learning-based approaches like DeepCubeA with traditional planning methods using PDDL and SAS+ representations could lead to generalization across the domain variations.

THANK YOU ALL

Contact Information

Bharath Muppasani –
bharath@email.sc.edu

LEARN MORE HERE!

AI4Society