Solving the Rubik's Cube with a PDDL Planner

Bharath Muppasani, Vishal Pallagani, Biplav Srivastava

AI Institute, University of South Carolina, Columbia, South Carolina, USA
{bharath@email., vishalp@email., biplav.s@}sc.edu

Abstract

Rubik's Cube (RC) is a popular puzzle that is also computationally hard to solve. In this demonstration, we introduce the first PDDL formulation for the 3x3 RC and solve it with an off-the-shelf Fast-Downward planner. Notably, we submitted this PDDL domain to the International Planning Competition (IPC) 2023's Classical track, where it emerged as one of the toughest domains to solve. We also create a plan executor and visualizer to show how the plan achieves the intended goal. Our system has two types of audiences: (a) planning researchers who can explore a hard problem and improve their planning algorithms, and (b) RC learners[1] who want to learn how to solve the puzzle at their own pace and can now modify an initial plan (e.g., manually, using other algorithms) and see their execution. See video at: https://youtu.be/YQZ2sj-x5js.

Introduction

As artificial intelligence (AI) continues to solve problems that humans struggle to solve, there is an emerging need for humans to understand these solutions so that we can trust AI, create new educational opportunities, and even discover new knowledge. Many of these problems are path-finding problems. That is, the problem is to find a sequence of actions (a path) to go from an initial state to a goal state. AI has been successfully applied to solve the Rubik's Cube (RC) (Agostinelli et al. 2019; Lakkaraju et al. 2022; Joyner 2008; Agostinelli et al. 2021) but these methods used opaque learning techniques which are hard for RC learners to benefit from.

While no PDDL encoding of a 3x3x3 RC problem is known to the authors, there is previous work[2] for a 2x2x2 RC setting and is solved with the Fast-Forward planner. Authors in (Büchner et al. 2022) modeled the RC problem in finite domain representation, which enables the common general purpose solvers to be used on the RC problem. Our contributions are:

- Introducing the first PDDL formulation for a 3x3x3 RC, a significant advancement that was subsequently submitted to the International Planning Competition (IPC) 2023.

[1]Individuals learning to solve the Rubik's Cube

[2]https://wu-kan.cn/2019/11/21/Planning-and-Uncertainty/

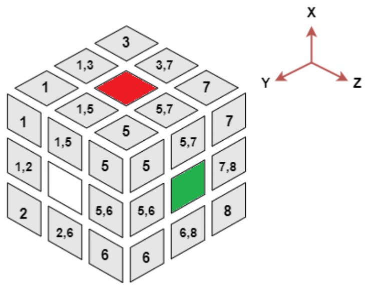


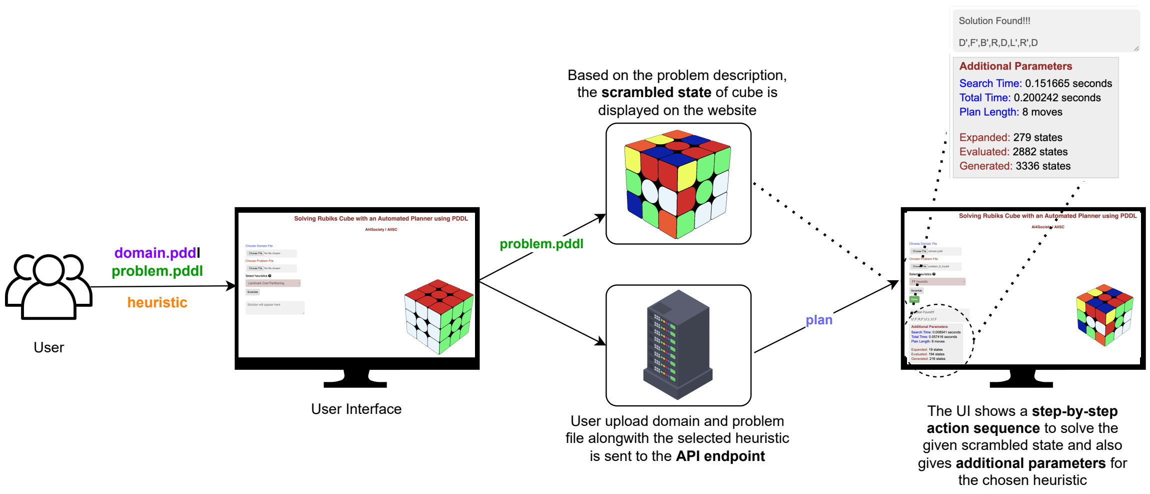
Figure 1: Rubik's cube description to define the domain encoding. For example, cube pieces 5, 6, 7, and 8 are in the right face

- Enabling RC learners to use off-the-shelf planners to find custom and optimized ways to solve any given RC configuration.

Moreover, for the learning-based RC solvers, which have been shown to scale to large instances, our PDDL model can be used as a labeled data generator for training. A demonstration can be seen at https://youtu.be/YQZ2sj-x5js.

System Description

RC representation in PDDL In the PDDL domain, the Rubik's cube problem environment has been defined by assuming the cube pieces are in a fixed position and are named accordingly. as defined in Figure 1. These fixed cube pieces are modeled as predicates in the RC domain and the colors they possess in the three-dimensional space as parameters of these predicates. With the help of conditional effects, each action in the RC environment is defined as the change of colors on these fixed cube pieces. The 3D axis of the cube is considered as three separate parameters X, Y, and Z that specify the position of the colors on the cube's pieces. One of these axes can be connected to each face of the cube. According to the representation shown in Figure 1, the respective faces on each axis are: $F_X = \langle U, D \rangle$; $F_Y = \langle F, B \rangle$; $F_Z = \langle R, L \rangle$. These different faces of the cube can be identified by the color of the middle cube piece. We considered White, Red, and Green colors as the colors on the front(F), up(U) and right(R) faces respectively (similarly, the counter colors on the counter faces).



domain.pddl
problem.pddl
heuristic
User
User Interface
problem.pddl
Based on the problem description, the scrambled state of cube is displayed on the website
plan
User upload domain and problem file alongwith the selected heuristic is sent to the API endpoint
Solution Found!!!
D',F',B',R,D,L',R',D
Additional Parameters
Search Time: 0.151665 seconds
Total Time: 0.200242 seconds
Plan Length: 8 moves
Expanded: 279 states
Evaluated: 2882 states
Generated: 3336 states
Solving Rubiks Cube with an Automated Planner using PDDL
The UI shows a step-by-step action sequence to solve the given scrambled state and also gives additional parameters for the chosen heuristic

Figure 2: System Architecture.

Listing 1: Action L of Rubik's Cube modeled in PDDL

```
(:action L
:effect (and
;for corner cubelets
(forall(?x ?y ?z)(when (cube1 ?x ?y ?z)
  (and (cube2 ?y ?x ?z))))
(forall(?x ?y ?z)(when (cube3 ?x ?y ?z)
  (and (cube1 ?y ?x ?z))))
(forall(?x ?y ?z)(when (cube4 ?x ?y ?z)
  (and (cube3 ?y ?x ?z))))
(forall(?x ?y ?z)(when (cube2 ?x ?y ?z)
  (and (cube4 ?y ?x ?z))))
;for edge cubelets
(forall(?x ?z)(when (edge13 ?x ?z)
  (and (edge12 ?x ?z))))
(forall(?y ?z)(when (edge34 ?y ?z)
  (and (edge13 ?y ?z))))
(forall(?x ?z)(when (edge24 ?x ?z)
  (and (edge34 ?x ?z))))
(forall(?y ?z)(when (edge12 ?y ?z)
  (and (edge24 ?y ?z))))))
```

Generating and Visualizing the Plan Our system's PDDL encoded RC solver, in combination with a Visualizer, generates plan actions to solve the problem using the Fast-Downward (FD) AI planner. We have used the publicly available three-rubiks-cube npm package [3] for 3D RC visualization. AI planners are controllable in generating the desired plans. We can specify the search algorithm and the heuristics to the Fast-Downward planner. Each different search algorithm generates different plans to reach the goal state. We employed A* search algorithm in combination with different heuristics which supports conditional-effects in our system and gives plans in minutes.

The system architecture is shown in Figure 2. The users need to upload the domain file[4] and the problem file of RC and select a heuristic of their choice from the dropdown menu. The uploaded domain file and problem file along with the selected heuristic are sent to the API endpoint. The RC visualizer is scrambled to match the initial state from the uploaded problem file. On the back end, the Fast-Downward Planner based on A* search along with the selected heuristic evaluates the uploaded problem and generates a solution. This solution is provided to the visualizer to solve the RC. The user may visually follow the actions in the plan file generated to observe the RC being solved step by step. Additional information on the parameters of search time, total time, evaluated states, expanded states, and generated states are also displayed in the front end.

Conclusion

In this work, we have demonstrated the capability of a planner to solve a complex puzzle, i.e., Rubik's Cube. For realizing this, we have created the first PDDL domain for RC. This approach not only facilitates a deeper understanding and interpretability compared to dedicated black-box solvers but also opens up new avenues for research in this field (Muppasani et al. 2023). In order to make the generated plan understandable by people outside the planning community as well, we have integrated the generated plan with a visualizer showing step-by-step moves to achieve a fully solved RC. In the future, we would like to perform a comparative study of the performance of various planners and different encodings (PDDL vs. SAS+) to solve a given RC configuration. Additionally, an empirical study on the performance of abstraction heuristics on the RC modeled in PDDL, which showed some promising results on SAS+ encoding of RC (Büchner et al. 2022), would be interesting. Integration of a suitable plan validator would be needed so that human-edited plans can be verified before execution (currently, VAL (Howey, Long, and Fox 2004) does not handle conditional effects). This would help us to assist a learner to solve RC under various constraints such as time or moves.

[3]https://github.com/lab89/three-rubiks-cube

[4]Available at IPC-2023 GitHub Repository

References

Agostinelli, F.; Mavalankar, M.; Khandelwal, V.; Tang, H.; Wu, D.; Berry, B.; Srivastava, B.; Sheth, A.; and Irvin, M. 2021. Designing Children's New Learning Partner: Collaborative Artificial Intelligence for Learning to Solve the Rubik's Cube. In *Interaction Design and Children*, 610–614.

Agostinelli, F.; McAleer, S.; Shmakov, A.; and Baldi, P. 2019. Solving the Rubik's cube with deep reinforcement learning and search. *Nature Mach. Intell.*, 1(8): 356–363.

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

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301. IEEE.

Joyner, D. 2008. *Adventures in group theory: Rubik's Cube, Merlin's machine, and other mathematical toys*. JHU Press.

Lakkaraju, K.; Hassan, T.; Khandelwal, V.; Singh, P.; Bradley, C.; Shah, R.; Agostinelli, F.; Srivastava, B.; and Wu, D. 2022. ALLURE: A Multi-Modal Guided Environment for Helping Children Learn to Solve a Rubik's Cube with Automatic Solving & Interactive Explanation. In *AAAI*.

Muppasani, B.; Pallagani, V.; Srivastava, B.; and Agostinelli, F. 2023. On Solving the Rubik's Cube with Domain-Independent Planners Using Standard Representations. *arXiv preprint arXiv:2307.13552*.