Cool Markovian Mathematical Framework for Rubik's Cube

alessandro1.barro@mail.polimi.it

March 2024

Introduction

In this article, we introduce a Markov Decision Process (MDP) framework specifically designed for the Rubik's Cube as a case study, fining application in Dynamic Programming (DP) and Reinforcement Learning (RL) tasks. We define a comprehensive methodology for representing states and executing manipulations through sequences of actions, while also providing a prototype algorithm to illustrate these concepts.

MDP overview

An MDP is a mathematical framework designed to model "memoryless" decision-making scenarios, best for situations where outcomes are partly random and partly under the control of a decision-maker. In our specific case, the transitions within the MDP are deterministic, offering an advantage in understanding and predicting system dynamics.

A MDP $\mathcal{M}$ can be described as the following tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle \tag{1}$$

Where $\mathcal{S}$ represents the state space, $\mathcal{A}$ denotes the action space, $\mathcal{R}$ embodies the reward function, $\mathcal{P}$ signifies the transition probability tensor, and γ is the discount factor. We shall now explore each of these components in detail, delineating the MDP denoted as $\mathcal{M}_R$.

State representation

We introduce an "open box" model that enables a 2-dimensional visualization of the cube. This approach presents a trade-off, necessitating the monitoring of the cube's spatial orientation and the adjacency of its components.

Face keys We define the face keys $\mathcal{F}$ as follows:

$$\mathcal{F}^{(t)} = \{F, B, U, D, L, R\}^{(t)} \tag{2}$$

Each key corresponds to the "front", "back", "up", "down", "left", and "right" faces of the cube. Although adjusting the cube's perspective will require a method, it's crucial to understand that these keys remain fixed consistently across each timestep $t \in \mathbb{N}$; notably each face key will have its center color piece constant aswell.

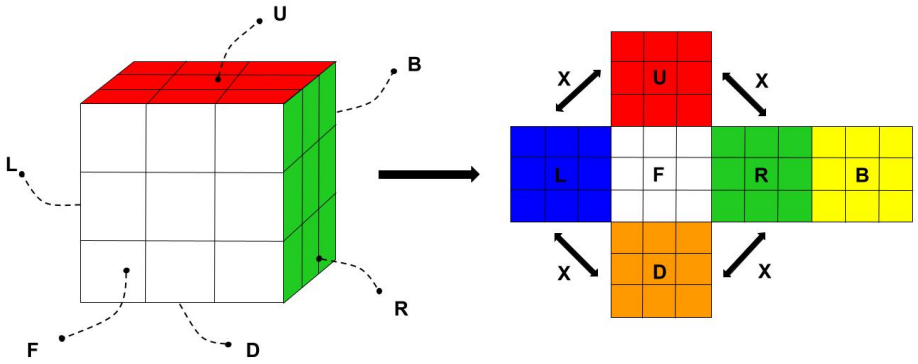

Figure 1: Open box model, unshuffled cube

To facilitate formalization and leverage intriguing properties, we conceptualize three planes corresponding to the face keys. Specifically, we have $F, B \in U$, $U, D \in V$, and $L, R \in W$. Consequently, $\mathcal{F}$ can be interpreted as the decomposition into three distinct subspaces:

$$\mathcal{F}^{(t)} = \{U, V, W\}^{(t)} \tag{3}$$

We will refer to a generic face as $K^{(t)}$, to which, at any time step $t \in \mathbb{N}$ a matrix f_i is allocated $K^{(t)} \leftarrow f_i$.

Value matrices As we just mentioned, to each individual key is assigned, at any iteration t, a 3×3 matrix f_i. If $\mathcal{K} = \{w, r, o, y, g, b\}$ is the color set, from which each element $f_{i_{lk}} \in f_i$ can take value, then a valid example would be:

$$L^{(2)} \leftarrow f_i = \begin{pmatrix} b & b & w \\ b & b & w \\ b & b & w \end{pmatrix} \tag{4}$$

Meaning that, to the back face L at step 2, is assigned the matrix f_i (this state is achieved by applying action $a = \langle U, + \rangle$ from an unshuffled cube).

Adjacency indicators Highlighting the spatial adjacency between specific faces is beneficial, particularly for those faces that lose this property in the open box model. We recognize that faces that lose their adjacency are the ones belonging to different planes lose, and faces within the same subspace are never adjacent. We will represent this relationship with the symbol $\mathbf{X}$, which is structured as follows:

$$\mathbf{X}_{\mathcal{F}_i \mathcal{F}_j} = \begin{cases} 1 & \text{if } \mathcal{F}_i \text{ is adjacent to } \mathcal{F}_j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

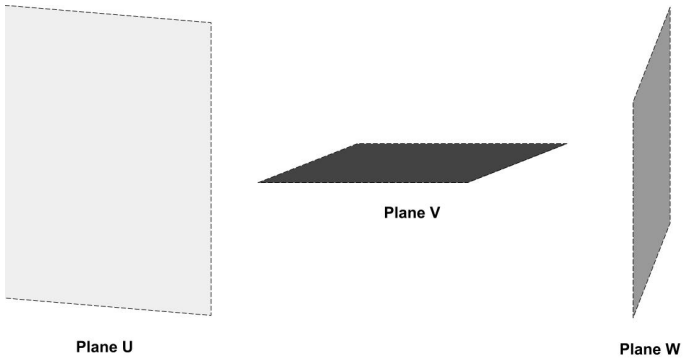

Figure 2: Planes U, V and W

An example indicator is $\mathbf{X}_{RU} = \mathbf{X}_{UR} = 1$. As face keys don't change position during iterations, not even adjacency indicators will.

We can finally define the state space as:

$$\mathcal{S} \equiv \mathcal{F}^{(t)} \tag{6}$$

Action modeling

Now for the interesting and less intuitive aspects. We formalize two types of actions: the first type involves rotating the cube without altering its configuration, effectively changing only the point of view (POV); the second type simulates executing a move that physically alters the cube's state, with rotations limited to the front face F due to our ability to switch POV. An action $a = \langle K, d \rangle$, where $d = \{+, -\}$ indicates "clockwise" and "counterclockwise" directions respectively, encapsulates both approaches. The strategy involves using POV adjustments to position the targeted face (or the matrix representing that face) in F, followed by executing the rotation. In either scenario, the action a transitions between time steps as follows:

$$a : \mathcal{F}^{(t)} \mapsto \mathcal{F}^{(t+1)} \tag{7}$$

To enhance understanding, we will apply these functions within the context of the "open box" state previously described, and individually specify the modifications to the matrices f_i. This approach avoids the complexity of a piece-wise function, which could lead to potentially excessive notation.

Prospective switches To manage the perspective change of the cube while maintaining logical precision, we acknowledge two potential directions for POV switching: horizontal h and vertical v. Considering the cube's state at time t, represented as $\mathcal{F}^{(t)} = \{f_2, f_4, f_5, f_6, f_1, f_3\}$, the process for a horizontal POV switch is outlined below:

$$\Lambda_h \left[\begin{pmatrix} & f_5 & & \\ f_1 & f_2 & f_3 & f_4 \\ & f_6 & & \end{pmatrix} \right] = \begin{pmatrix} & f_5' & & \\ f_2 & f_3 & f_4 & f_1 \\ & f_6' & & \end{pmatrix} \tag{8}$$

In this framework, given the 3-dimensional rotation matrix $\Pi \in \mathbb{R}^{3\times3}$, the transformations for f_5 and f_6 during the horizontal switch are defined as:

$$f_5' = \Pi\left(\frac{\pi}{2}\right) f_5 \tag{9}$$

$$f_6' = \Pi\left(-\frac{\pi}{2}\right) f_6 \tag{10}$$

More broadly, for any face f_i, the transformation can be generalized as:

$$f_i' = \Pi(\alpha) f_i \tag{11}$$

Here, α denotes the rotation angle, with possible values $\{\frac{\pi}{2}, -\frac{\pi}{2}\}$, indicating clockwise or counterclockwise rotation, respectively.

It's important to note that while keys in the V subspace are rotated as described, keys in the W subspace undergo translation. However, the specific formulation for this translation action is not provided here for simplicity.

The vertical switch operates as follows, offering a different manipulation of the cube's faces compared to the horizontal switch:

$$\Lambda_v \left[\begin{pmatrix} & f_5 & & \\ f_1 & f_2 & f_3 & f_4 \\ & f_6 & & \end{pmatrix} \right] = \begin{pmatrix} & f_4'' & & \\ f_1' & f_5 & f_3' & f_6'' \\ & f_2 & & \end{pmatrix} \tag{12}$$

For this transformation, the specific face modifications are defined as:

$$f_1' = \Pi\left(\frac{\pi}{2}\right) f_1 \tag{13}$$

$$f_3' = \Pi\left(-\frac{\pi}{2}\right) f_3 \tag{14}$$

$$f_4'' = \Pi(\pi) f_5 \tag{15}$$

$$f_6'' = \Pi(\pi) f_6 \tag{16}$$

In a more generalized form, for any given face f_i, the transformation can be represented as:

$$f_i'' = \Pi(\beta) f_i \tag{17}$$

Here, β can take the values $\{-\pi, \pi\}$, which denote the specific rotation angles for the transformation Π. This means a rotation of π for some faces, indicating a complete inversion, while the ones remaining on the W subspace are rotated by $\frac{\pi}{2}$ to align with the new vertical orientation.

Rotations To elaborate on the rotational dynamics within the model, we focus on the interactions with individual matrix values for each face, f_i, while keeping the global arrangement of the keys' matrix assignment constant. The operation Γ_+ signifies a clockwise rotation, impacting the cube as follows:

$$\Gamma_+ \left[\begin{pmatrix} & f_5 & & \\ f_1 & f_2 & f_3 & f_4 \\ & f_6 & & \end{pmatrix} \right] = \begin{pmatrix} & f_5^* & & \\ f_1^* & f_2' & f_3^* & f_4 \\ & f_6^* & & \end{pmatrix} \tag{18}$$

In this setup, f_2' undergoes a direct rotation defined by:

$$f_2' = \Pi \left(\frac{\pi}{2} \right) f_2, \tag{19}$$

The transformations of the adjacent faces—f_1, f_3, f_5, and f_6—result from the rotation of f_2 and are expressed as:

$$f_3^* = \mathbf{X}_{RU} \mathbf{R}_{\text{III}} \oplus \mathbf{C}_{\text{II}} \oplus \mathbf{C}_{\text{III}} \in W, \tag{20}$$
$$f_6^* = \mathbf{X}_{DR} \mathbf{C}_{\text{III}} \oplus \mathbf{R}_{\text{II}} \oplus \mathbf{R}_{\text{III}} \in V, \tag{21}$$
$$f_1^* = \mathbf{C}_{\text{I}} \oplus \mathbf{C}_{\text{II}} \oplus \mathbf{X}_{LD} \mathbf{R}_{\text{I}} \in W, \tag{22}$$
$$f_5^* = \mathbf{R}_{\text{I}} \oplus \mathbf{R}_{\text{II}} \oplus \mathbf{X}_{UL} \mathbf{C}_{\text{III}} \in V. \tag{23}$$

The symbol "$\oplus$" denotes matrix assembly from selected rows or columns, reflecting the new orientation of cube elements post-rotation. The transformation employs the previously defined adjacency operators ($\mathbf{X}_{RU}$, $\mathbf{X}_{DR}$, etc.) to map rows or columns from their original positions to new ones.

The generalized transformation for any face f_i affected by the rotation is captured by:

$$f_i^* = \bigoplus_{l=1}^{3} \mathbf{X}_l \overline{\mathbf{H}}_l(f_j) \bigoplus_{m \neq l} \mathbf{H}_m(f_i), \tag{24}$$

where $\mathbf{H} = \{\mathbf{R}, \mathbf{C}\}$ denotes the set of rows or columns, and, interestingly enough, this is contingent on whether f_i belongs to subspace V or W. Of course, f_j indicated f_i's adjacent matrix $d =$ " $+$ " direction.

For counter-clockwise rotations, represented by Γ_-, the procedure mirrors that of Γ_+ but in the reverse direction.

Actions With the foundational components in place, we can now define the action space of our Rubik's Cube model:

$$\mathcal{A} \equiv \{\Lambda, \Gamma\} \tag{25}$$

Here, Λ represents the POV switching actions, which can be either horizontal (Λ_h) or vertical (Λ_v), and Γ signifies the rotational actions, which can be

either clockwise (Γ_+) or counterclockwise (Γ_-) rotations of the cube's faces. In particular, as previously denoted:

$$a = \langle K, d \rangle \tag{26}$$

Executing an action a typically involves a sequential application of both Λ and Γ. It's uncommon to apply either action in isolation due to the nature of the cube's manipulation. For example, the action tuple $a = \langle R, + \rangle$, which denotes rotating the right face clockwise, necessitates initially performing a horizontal POV switch (Λ_h) to align the right face as the new front face (F). This reorientation is followed by the clockwise rotation (Γ_+) applied to the now-front face.

The combination of POV switching and rotational actions allows for a comprehensive and efficient exploration of the cube's state space, ensuring that any desired, new configuration can be reached through a series of well-defined actions.

Algorithm 1 Performing action $a = \langle K, d \rangle$

input $K \in \mathcal{F}^{(t)}$, $d = \{+, -\}$
$K \rightarrow f_K$
if $K \in V$ **then**
  while $F \not\leftarrow f_k$ **do**
    $\mathcal{F}^{(t+1)} = \Lambda_v[\mathcal{F}^{(t)}]$
  end while
  $\mathcal{F}^{(T)} = \Gamma_d[\mathcal{F}^{(T-1)}]$
else if $K \in U \vee K \in W$ **then**
  while $F \not\leftarrow f_k$ **do**
    $\mathcal{F}^{(t+1)} = \Lambda_h[\mathcal{F}^{(t)}]$
  end while
  $\mathcal{F}^{(T)} = \Gamma_d[\mathcal{F}^{(T-1)}]$
end if

Reward function

Constructing an effective reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ presents a significant challenge, especially when it comes to evaluating the disorder of a state $\mathcal{F}^{(t)}$ at any given iteration. Utilizing a metric that assesses the number of moves from a solution determined by a conventional solving algorithm, might induce a bias towards existing solution strategies, rather than encouraging the discovery of new methods.

To avoid this, a more desirable approach would focus on evaluating states independently of the actions taken to reach them. So, we should define a 'state distance' metric that quantifies the deviation of a current state $s = \mathcal{F}^{(t)}$ from

the solved state $s_{\text{end}} = \mathcal{F}^{(T)}$. This could be formally expressed as:

$$\mathcal{R}(s, a) = -||s' - s_{\text{end}}||_R \tag{27}$$

where $||\cdot||_R$ symbolizes a distance measure within the state space, and is not to be confused with a conventional vector norm. In fact, this distance metric might be non-linear, reflecting the relationship between cube states and the number of moves needed to solve the cube, which often follows an exponential pattern.

One straightforward approach could involve quantifying the 'correctness' of individual cube faces. Given that the center piece of each face remains constant, the reward function could account for the number of edge and corner pieces correctly positioned relative to these fixed centers. This count could then be incorporated into a non-linear function, such as an exponential, to calculate the reward:

$$\mathcal{R}(s, a) = -\exp\left(-\sum_{k \in \mathcal{F}} C_k(s)\right) \tag{28}$$

Here, $C_k(s)$ represents the count of correctly positioned pieces for face k in state s, and the sum runs over all faces in the set of face keys $\mathcal{F}$. The exponential function amplifies the impact of each correctly positioned piece, with the negative sign ensuring that a lower 'distance' (i.e., a higher number of correct pieces) yields a higher reward.

Incorporating an epsilon-greedy strategy within this framework can enhance exploration, particularly in the initial stages of learning. This method would allow the model to explore a diverse set of actions early on, gradually shifting towards more deterministic actions as it learns the structure of the reward landscape.

Transition probabilities

In the context of our MDP, due to the deterministic nature of the Rubik's Cube mechanics, each action $a \in \mathcal{A}$ taken in a state $s \in \mathcal{S}$ leads to a predetermined subsequent state $s' \in \mathcal{S}$. This characteristic simplifies the transition probability tensor significantly:

$$\mathcal{P}(s'|s, a) = \begin{cases} 1 & \text{if action } a \text{ in state } s \text{ leads to state } s', \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

This binary nature of the transition probabilities implies that the outcome of any action is entirely predictable, eliminating the uncertainty typically associated with stochastic environments.

MDP formulation

In this article, we have assembled the components necessary to construct a Markov Decision Process (MDP) tailored to the Rubik's Cube, denoted as $\mathcal{M}_R$. This MDP is defined by the tuple:

$$\mathcal{M}_R = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle_R, \tag{30}$$

The discount factor γ is pivotal in guiding the solution strategy. Given the structure of the Rubik's Cube, where a seemingly low-reward state can be a precursor to a highly rewarding configuration, a strategy leaning towards a greedy approach might and might not be optimal. Instead, a balanced or dynamic adjustment of γ could better navigate the complex landscape of the state space. With this work, we would like to extend beyond puzzle solving, offering insights into the algorithmic manipulation of complex systems and the strategic exploration of deterministic state spaces leveraging techniques such as DP and RL.