Cool Markovian Framework for Rubik's Cube

alessandro1.barro@mail.polimi.it

March 2024

Introduction

In this article, we show a Markov Decision Process (MDP) framework tailored for analyzing and solving the Rubik's Cube, finding its potential utility in the realms of Dynamic Programming (DP) and Reinforcement Learning (RL). Our approach delineates a methodology for the representation of the cube's varying states and the execution of manipulative actions that alter these states. We later advance in the definition of a reward path and agent exploration strategies, ultimating the decision process' landscape.

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

In our framework, we employ an "open box" model to represent the Rubik's Cube, which allows us to depict the cube's complex 3D structure in a simplified 2D format. This model unfolds the cube in a way that all faces are visible at once, akin to unfolding a box so that each side lies flat. While this visualization technique offers the advantage of easily observable state changes on a 2D plane, it introduces the challenge of keeping track of the cube's 3D orientation and the spatial relationships between different faces.

Keys and Planes We define the cube's face keys as follows:

$$\{F, B, U, D, L, R\} \tag{2}$$

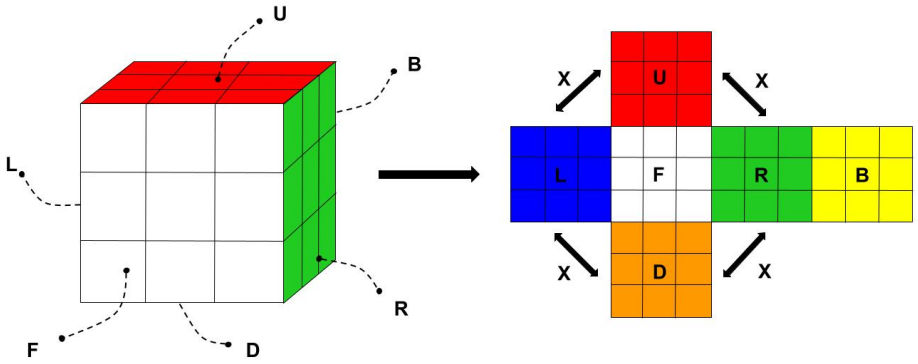

Figure 1: Open box model, unshuffled cube

Each key corresponds to the front, back, up, down, left, and right faces of the cube; from this point on, we will often refer to a generic i-th key. It's crucial to understand that these keys remain fixed consistently across each iteration timestep $t \in \mathbb{N}$. Notably each face key will have its center color piece constant aswell.

For mathematical formalization purposes and for exploiting the unique characteristics of the Rubik's Cube, we conceptualize three orthogonal planes, each aligned with specific faces of the cube. In particular, $F, B \in \mathbf{U}$, $U, D \in \mathbf{V}$, and $L, R \in \mathbf{W}$. Mathematically, this division can be represented as:

$$\mathbf{F} = \mathbf{U} \oplus \mathbf{V} \oplus \mathbf{W} \tag{3}$$

where $\oplus$ symbolizes the direct sum, underscoring the orthogonal nature of the subspaces within the cube's geometric configuration.

Value matrices We define $\mathcal{K} = \{w, r, o, y, g, b\}$ to be the color set. At any iteration $t = 1, \ldots, T$, each cube's face is modelized as a matrix $\mathbf{f}_i \in \mathcal{K}^{3\times 3}$.Then a valid example would be:

$$\mathbf{f}_L^{(2)} = \begin{pmatrix} b & b & w \\ b & b & w \\ b & b & w \end{pmatrix} \tag{4}$$

Meaning that, the left face after one move is represented by the matrix $\mathbf{f}_L^{(2)}$ (this state is achieved by applying action $a = \langle U, + \rangle$ from an unshuffled cube, which possesses a specific initial color configuration). We would like to collect each value matrix being coherent with each other at a certain time step t as follows:

$$\mathbf{F}^{(t)} = \{\mathbf{f}_F, \mathbf{f}_B, \mathbf{f}_U, \mathbf{f}_D, \mathbf{f}_L, \mathbf{f}_R\}^{(t)} \tag{5}$$

Adjacency indicators To capture the adjacency relationships between cube faces, particularly those disrupted in the 2D "open box" representation, we

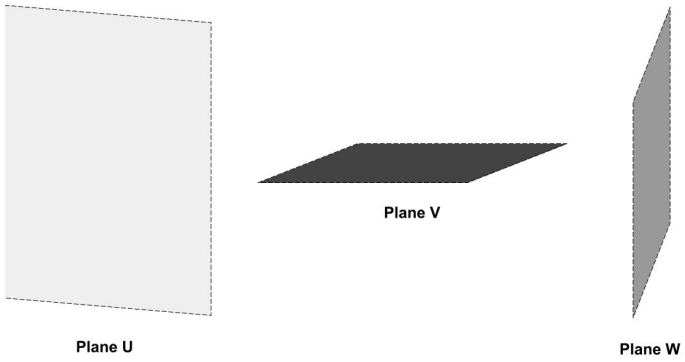

Figure 2: Planes **U**, **V** and **W**

employ adjacency matrices denoted by **X**. In this model, adjacency is preserved within the same plane but lost across different planes. The adjacency matrix **X** is defined as:

$$\mathbf{X}_{ij} = \begin{cases} 1 & \text{if } i \text{ is adjacent to } j \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Each element will serve as useful indicators for the states manipulation.

We can finally define the state space as:

$$\mathcal{S} \equiv \mathbf{F}^{(t)} \tag{7}$$

Action modeling

We define two distinct categories of actions within our model: the first alters the cube's orientation without directly changing the values in the matrices $\mathbf{f}_i$, while the second modifies the $\mathbf{f}_F$ matrix values, impacting the cube's state without affecting its overall orientation. This dichotomy allows for comprehensive control over the cube's configuration. An action $a = \langle i, d \rangle$, where d represents the direction of rotation, + for clockwise (CW) and - for counterclockwise (CCW), integrates these two action types. The execution of any action a transitions the cube from one state to the next across consecutive time steps.

$$a : \mathbf{F}^{(t)} \mapsto \mathbf{F}^{(t+1)} \tag{8}$$

Orientations To manage the perspective change of the cube while maintaining logical precision, we acknowledge two potential directions for switching: horizontal h and vertical v. The process for a horizontal switch Φ_h is outlined below:

$$\Phi_h \left[\begin{pmatrix} & \mathbf{f}_U & & \\ \mathbf{f}_L & \mathbf{f}_F & \mathbf{f}_R & \mathbf{f}_B \\ & \mathbf{f}_D & & \end{pmatrix}^{(t)} \right] = \begin{pmatrix} & \phi(\mathbf{f}_U) & & \\ \mathbf{f}_F & \mathbf{f}_R & \mathbf{f}_B & \mathbf{f}_L \\ & \phi(\mathbf{f}_D) & & \end{pmatrix}^{(t)} \tag{9}$$

Within our model, the operation ϕ represents a single step of rotating a matrix $\mathbf{f}_i$. This rotation can be either CW or CCW. For any two matrices $\mathbf{f}_i, \mathbf{f}_j$ within the same subspace $\mathbf{V}$, if $\phi(\mathbf{f}_i)$ is executed as a CW rotation, then $\phi(\mathbf{f}_j)$ must be executed as a CCW rotation to maintain the system's consistency. The operation ϕ can be viewed as a combination of matrix transposition and permutation:

$$\phi(\mathbf{f}_i) = (\mathbf{f}_i)^\top \mathbf{J}, \tag{10}$$

where $\mathbf{J}$ is a 3×3 matrix characterized by ones on its anti-diagonal, facilitating a permutation of columns subsequent to transposition. For a CCW rotation, the permutation is applied to rows, as shown in:

$$\phi(\mathbf{f}_j) = \mathbf{J}(\mathbf{f}_j)^\top. \tag{11}$$

It's important to note that while matrices in the $\mathbf{V}$ subspace are subject to rotations as described above, matrices in the $\mathbf{U}, \mathbf{W}$ subspace experience translation from one subspace to the other (e.g. $\mathbf{f}_F^{(t+1)} = \mathbf{f}_R^{(t)}$). In general, we can state that:

$$\Phi_h(\mathbf{f}_i) = \begin{cases} \mathbf{f}_j \in \mathbf{W} & \text{if } \mathbf{f}_i \in \mathbf{U} \wedge \mathbf{X}_{ij} = 1 \\ \phi(\mathbf{f}_i) & \text{if } \mathbf{f}_i \in \mathbf{V} \\ \mathbf{f}_j \in \mathbf{U} & \text{if } \mathbf{f}_i \in \mathbf{W} \wedge \mathbf{X}_{ij} = 1 \end{cases} \tag{12}$$

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

$$\Phi_v \left[\begin{pmatrix} & \mathbf{f}_U & & \\ \mathbf{f}_L & \mathbf{f}_F & \mathbf{f}_R & \mathbf{f}_B \\ & \mathbf{f}_D & & \end{pmatrix}^{(t)} \right] = \begin{pmatrix} & \phi''(\mathbf{f}_B) & & \\ \phi(\mathbf{f}_L) & \mathbf{f}_U & \phi(\mathbf{f}_R) & \phi''(\mathbf{f}_D) \\ & \mathbf{f}_F & & \end{pmatrix}^{(t)} \tag{13}$$

Where ϕ'' is the two-times recursion of ϕ, resulting in:

$$\phi''(\mathbf{f}_i) = \phi(\phi(\mathbf{f}_i) = (\mathbf{f}_i^\top \mathbf{J})^\top \mathbf{J} = \mathbf{J}^\top \mathbf{f}_i \mathbf{J} \tag{14}$$

This is equivalent to a complete inversion or face flip. The faces on the $\mathbf{W}$ subspace are rotated one time to align with the new vertical orientation.

$$\Phi_v(\mathbf{f}_i) = \begin{cases} \phi''(\mathbf{f}_j) & \text{if } \mathbf{f}_i \in \mathbf{U} \wedge \mathbf{X}_{ij} = 1 \\ \phi''(\mathbf{f}_j) & \text{if } \mathbf{f}_i \in \mathbf{V} \wedge \mathbf{X}_{ij} = 1 \\ \phi(\mathbf{f}_i) & \text{if } \mathbf{f}_i \in \mathbf{W} \end{cases} \tag{15}$$

This notation is simplified, as we need to take into account the opposite direction in single rotations ϕ for faces in $\mathbf{W}$; while acknowledging that flipping ϕ'' applies in transitions between $\mathbf{U}$ and $\mathbf{V}$ only for U, B keys.

Moves To elaborate on the cube editing dynamics within the model, we focus on the paired interactions between matrix values, $\mathbf{f}_i, \mathbf{f}_j$, while keeping the global orientation practically constant. The operation Ψ_+ signifies a clockwise rotation of F key, impacting the cube as follows:

$$\Psi_+ \left[\begin{pmatrix} & \mathbf{f}_U & & \\ \mathbf{f}_L & \mathbf{f}_F & \mathbf{f}_R & \mathbf{f}_B \\ & \mathbf{f}_D & & \end{pmatrix}^{(t)} \right] = \begin{pmatrix} & \psi(\mathbf{f}_U, \mathbf{f}_L) & & \\ \psi(\mathbf{f}_L, \mathbf{f}_D) & \phi(\mathbf{f}_F) & \psi(\mathbf{f}_R, \mathbf{f}_U) & \mathbf{f}_B \\ & \psi(\mathbf{f}_D, \mathbf{f}_R) & & \end{pmatrix}^{(t)} \tag{16}$$

In this setup, $\mathbf{f}_2$ undergoes a direct rotation which was previously defined. To come close to the definition of ψ, we outline the transformations of the adjacent faces—$\mathbf{f}_1$, $\mathbf{f}_3$, $\mathbf{f}_5$, and $\mathbf{f}_6$—resulted from the rotation of $\mathbf{f}_2$:

$$\mathbf{f}_R^{(t+1)} = \mathbf{R}_{\mathrm{III}}(\mathbf{f}_U^{(t)}) \oplus \mathbf{C}_{\mathrm{II}}(\mathbf{f}_R^{(t)}) \oplus \mathbf{C}_{\mathrm{III}}(\mathbf{f}_R^{(t)}) \tag{17}$$

$$\mathbf{f}_D^{(t+1)} = \mathbf{C}_{\mathrm{III}}(\mathbf{f}_R^{(t)}) \oplus \mathbf{R}_{\mathrm{II}}(\mathbf{f}_D^{(t)}) \oplus \mathbf{R}_{\mathrm{III}}(\mathbf{f}_D^{(t)}) \tag{18}$$

$$\mathbf{f}_L^{(t+1)} = \mathbf{C}_{\mathrm{I}}(\mathbf{f}_L^{(t)}) \oplus \mathbf{C}_{\mathrm{II}}(\mathbf{f}_L^{(t)}) \oplus \mathbf{X}_{LD}\mathbf{R}_{\mathrm{I}}(\mathbf{f}_D^{(t)}) \tag{19}$$

$$\mathbf{f}_U^{(t+1)} = \mathbf{R}_{\mathrm{I}}(\mathbf{f}_U^{(t)}) \oplus \mathbf{R}_{\mathrm{II}}(\mathbf{f}_U^{(t)}) \oplus \mathbf{C}_{\mathrm{III}}(\mathbf{f}_L^{(t)}) \tag{20}$$

In this case, the symbol $\oplus$ denotes matrix assembly from selected rows or columns, reflecting the new orientation of cube elements post-rotation. The generalized transformation for any face $\mathbf{f}_i$ affected by the rotation is captured by:

$$\psi(\mathbf{f}_i, \mathbf{f}_j) = \overline{\mathbf{H}}_l(\mathbf{f}_j) \bigoplus_{m \neq l} \mathbf{H}_m(\mathbf{f}_i), \tag{21}$$

where $\mathbf{H} = \{\mathbf{R}, \mathbf{C}\}$ denotes the set of rows or columns, and, interestingly enough, this is contingent on whether $\mathbf{f}_i$ belongs to subspace $\mathbf{V}$ or $\mathbf{W}$. Of course, $\mathbf{f}_j$ indicates $\mathbf{f}_i$'s adjacent matrix in CW direction. A generic formula would be:

$$\Psi_+(\mathbf{f}_i) = \begin{cases} \phi(\mathbf{f}_i) & \text{if } i = F \\ \mathbf{f}_i & \text{if } i = B \\ \psi(\mathbf{f}_i, \mathbf{f}_j) & \text{if } \mathbf{f}_i \in \mathbf{V}, \mathbf{W} \wedge \mathbf{X}_{ij} = 1 \end{cases} \tag{22}$$

Again, this reflects a simplistic approach as the nature of ψ itself precludes a sense of direction in the choice of the adjacent j-th key. For counter-clockwise rotations, represented by Ψ_-, the procedure mirrors that of Ψ_+ 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:

$$\mathcal{A} \equiv \{\Phi, \Psi\} \tag{23}$$

Here, Φ represents the orientation switching actions, which can be either horizontal (Ψ_h) or vertical (Φ_v); for these two, it's essential to establish a direction for the transitioning faces aswell. Ψ signifies the rotational actions, which can be either clockwise (Ψ_+) or counterclockwise (Ψ_-) rotations of the cube's faces. In particular, as previously denoted:

$$a = \langle i, d \rangle \tag{24}$$

Executing an action a typically involves a sequential application of both Φ and Ψ. 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 switch (Φ_h) to align the right face as the new front face (F). This reorientation is followed by the clockwise rotation (Ψ_+) applied to the now-front face.

The combination of orientation 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 $a = \langle i, d \rangle$

input i, d
if $i \in \mathbf{V}$ **then**
  while $\mathbf{f}_F \not\leftarrow \mathbf{f}_i$ **do**
    $\mathbf{F}^{(t+1)} = \Psi_v[\mathbf{F}^{(t)}]$
  end while
  $\mathbf{F}^{(T)} = \Psi_d[\mathbf{F}^{(T-1)}]$
else if $i \in \mathbf{U} \vee i \in \mathbf{W}$ **then**
  while $\mathbf{f}_F \not\leftarrow \mathbf{f}_i$ **do**
    $\mathbf{F}^{(t+1)} = \Phi_h[\mathbf{F}^{(t)}]$
  end while
  $\mathbf{F}^{(T)} = \Psi_d[\mathbf{F}^{(T-1)}]$
end if

Reward function

Constructing an effective reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ 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:

$$\mathcal{R}(s, a) = -||s' - s_{\text{end}}||_R \tag{25}$$

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:

$$||s' - s_{\text{end}}||_R = \exp\left(-\sum_{k \in \mathcal{F}} C_k(s)\right) \tag{26}$$

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:

$$\mathcal{P}(s'|s, a) = \begin{cases} 1 & \text{if action } a \text{ in state } s \text{ leads to state } s', \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

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:

$$\mathcal{M}_R = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle_R, \tag{28}$$

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.