UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO RIBEIRO MONTEIRO

Generalização do problema do Cubo Mágico

RIO DE JANEIRO
2020

GUSTAVO RIBEIRO MONTEIRO

Generalização do problema do Cubo Mágico

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Daniel Sadoc Menasché

RIO DE JANEIRO
2020

CIP - Catalogação na Publicação

M775g

Monteiro, Gustavo Ribeiro
Generalização do problema do Cubo Mágico / Gustavo Ribeiro Monteiro. -- Rio de Janeiro, 2020.
46 f.

Orientador: Daniel Sadoc Menasché.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2020.

1. Cubo Mágico. 2. Rubik's Cube. 3. Generalização. I. Menasché, Daniel Sadoc, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

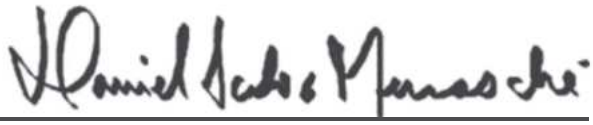
GUSTAVO RIBEIRO MONTEIRO

Generalização do problema do Cubo Mágico

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 28 de julho de 2020

BANCA EXAMINADORA:



Daniel Sadoc Menasche
Professor Adjunto (UFRJ)

Documento assinado digitalmente
gov.br JOAO ANTONIO RECIO DA PAIXAO
Data: 10/04/2024 19:26:46-0300
Verifique em https://validar.iti.gov.br

João Antônio Recio da Paixão
Professor Adjunto (UFRJ)

Documento assinado digitalmente
gov.br JOAO CARLOS PEREIRA DA SILVA
Data: 10/04/2024 08:31:15-0300
Verifique em https://validar.iti.gov.br

João Carlos Pereira da Silva
Professor Associado (UFRJ)

Dedico esse trabalho a todos que entendem que a certeza é uma ilusão.

AGRADECIMENTOS

Agradeço ao estado brasileiro que financia e tem o ensino superior como parte fundamental da nação. Agradeço à Universidade Federal do Rio de Janeiro que me proporcionou direta e indiretamente diversas oportunidades para desenvolver os meus conhecimentos.

Agradeço a todos os professores e professoras que tive ao longo de minha trajetória e que criaram toda a base necessária para eu seguir construindo meu conhecimento. Agradeço em especial ao meu orientador Daniel Sadoc que aceitou este desafio mesmo sem eu ter cursado suas disciplinas.

Agradeço também aos meus colegas e amigos de faculdade que me proporcionaram longos debates sobre diversos assuntos ligados ou não à computação, e que me ajudaram a ter uma visão mais plural do conhecimento, o que foi fundamental para fazer esse trabalho.

Agradeço aos amigos de cubo mágico, que me apresentaram esse imenso mundo dos quebra-cabeças 3D e despertaram meu interesse e curiosidade a ponto de me fazer querer entender mais a fundo.

Agradeço também à família, aos amigos de longas datas, e às namoradas que passaram na minha vida, pois sem esse grupo de pessoas, a vida não teria o sabor que tem e jamais conseguiria fechar esse e tantos outros trabalhos.

"If I have seen further
it is by standing on the shoulders of Giants."

Isaac Newton

RESUMO

O Cubo Mágico vem desde os anos 80 encantando pessoas por misturar uma simplicidade aparente com uma complexidade desafiadora. O cubo originalmente é um exemplo de um problema com muitos casos possíveis e apenas um caso desejado. Ao longo do tempo foram criados diversos métodos para resolver esse quebra cabeça. Alguns métodos foram feitos para humanos, o que permitiu o chinês Yusheng Du bater o record e resolver o cubo em 3,47 segundos em 2018. Já outros métodos foram feitos para máquinas, permitindo que pesquisadores descobrissem que qualquer estado do cubo possa ser resolvido em até 20 movimentos. O presente trabalho tem como objetivo expandir o problema do Cubo Mágico e criar um meio eficiente de levar um estado qualquer do cubo a qualquer outro estado do cubo. Para isso utilizaremos uma rede neural já treinada chamada DeepCubeA que de forma rápida consegue resolver o cubo gerando soluções curtas e por vezes soluções ótimas.

Palavras-chave: Cubo Mágico; Rubik's Cube; Generalização.

ABSTRACT

Since the 80s, the Rubik's Cube has been involving people for mixing apparent simplicity with challenging complexity. The Cube is a good example of a problem that has multiple possible states and only one desired state. Many techniques for solving the puzzle were developed through time. Some of those were developed for humans which permitted the chinese Yusheng Du to beat the record solving the Rubik's cube in only 3,47 seconds. And other techniques were developed for machines, which made researchers discover that any cube state can be solved in until 20 moviments. This work has the goal to expand the cube's problem and develop an efficient way to go from any state to another. For that, we use a trained Artificial Neural Network called DeepCubeA which is capable of solving any state of Rubik's Cube fastly and, most of the time, returning the best solution.

Keywords: Rubik's Cube; Generalization.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	Artificial Neural Network
BWAS	Batch Weighted A* Search

LISTA DE SÍMBOLOS

λ	Lambda
θ	Theta

SUMÁRIO

1 INTRODUÇÃO

O Rubik's Cube ou Cubo Mágico foi criado em 1974 pelo professor de arquitetura Erno Rubik e teve grande popularidade e reconhecimento nos anos 80. O fascínio do quebra-cabeça se dá pela sua simplicidade aparente, que mascara uma grande complexidade para atingir seu estado resolvido.

Intuitivamente, podemos pensar o Cubo Mágico (cubo) como sendo construído a partir de 27 cubinhos organizados em formato de um cubo com dimensões 3x3x3. Apenas 26 cubinhos são visíveis e cada face aparente dos cubinhos é adesivada com uma de 6 cores. Um estado do cubo pode ser entendido como uma configuração única das cores pelo cubo. Diz-se que o cubo está em seu estado resolvido quando cada uma de suas faces possui apenas uma cor. Para levar o cubo de um estado a outro, é possível rotacionar em sentido horário ou anti-horário em 90^{o} ou 180^{o} cada grupo de 9 cubinhos coplanares do cubo.

Com esse simples conjunto de regras, o cubo possui apenas um estado resolvido entre cerca de $4,3.10^{19}$ estados possíveis. Sendo assim, o cubo se torna promissor para o desenvolvimento de técnicas de como lidar com problemas de muitos casos possíveis e poucos casos desejados.

Ao longo dos anos, muitos métodos foram desenvolvidos para resolver o problema de levar qualquer configuração do cubo para o estado resolvido. Alguns desses métodos foram desenvolvidos para serem executados por humanos, já outros foram desenvolvidos para serem executados por máquinas.

Uma equipe de pesquisadores da Califórnia publicou em 2019 uma rede neural profunda chamada DeepCubeA. Essa rede é capaz de levar qualquer estado do cubo para o estado resolvido. Em cerca de 60% dos casos, segundo os dados de teste, a rede gera o caminho com a menor quantidade de movimentos possível (AGOSTINELLI et al., 2019).

A DeepCubeA se baseia na busca A*, que por sua vez demanda uma função heurística admissível para garantir resultados ótimos. Como seria impraticável gerar uma tabela de custo de transição para cada estado do cubo, a estratégia adotada foi criar uma rede neural artificial que oferecesse o valor da heurística para cada estado. A abordagem não garante uma heurística admissível como outras abordagens garantem, porém é a rede entrega resultados com um consumo de processamento e memória relativamente baixos além de ter seu código disponível e de fácil acesso[1].

[1] O código original do DeepCubeA em python2 pode ser encontrado em https://codeocean.com/capsule/5723040/tree/v1

1.1 MOTIVAÇÃO

A solução por rede neural artificial, e muitas outras soluções, resolve o problema de múltiplas origens para um destino. Ou seja, leva qualquer um dos cerca de 4.10^{19} estados possíveis do cubo, para o único estado resolvido. Porém, a rede não resolve o problema de múltiplas origens para múltiplos destinos.

Existem abordagens que podem ser utilizadas para resolver o problema geral e levar um estado qualquer do cubo a outro estado qualquer do cubo, porém não é do nosso conhecimento alguma técnica que permita resolver esse problema de forma ótima com o custo de processamento equivalente ao resolver o problema original.

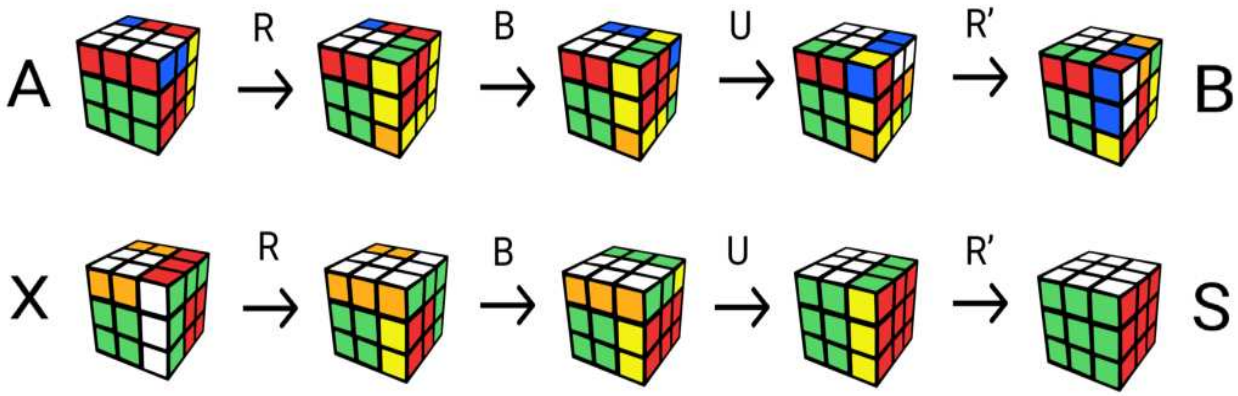
1.2 CONTRIBUIÇÃO

1.2.1 Resolver o problema geral do cubo mágico

O principal objetivo deste trabalho é expandir o problema tradicional do cubo de múltiplas origens e um destino, para um problema geral de múltiplas origens e múltiplos destinos sem *overhead* computacional significativo. Ou seja, criar um método para levar qualquer estado do cubo a qualquer outro estado do cubo com custo computacional semelhante ao resolver o problema tradicional do cubo mágico.

A ideia se baseia na independência entre estados e movimentos. Exemplo: considerando o estado resolvido **S** (com uma cor para cada face) e 2 estados quaisquer do cubo **A** e **B**. Se formos capazes de criar um estado auxiliar **X** tal que a sequência de movimentos que leva de **X** para **S** seja a mesma sequência de movimentos que leva de **A** para **B**, então podemos usar as técnicas já conhecidas para resolver de **X** para **S**, e assim encontramos a sequência de movimentos que leva do estado **A** para o estado **B**.

Figura 1 – Exemplo simples em que aplicar a mesma sequência de movimentos leva o cubo de **A** para **B** e também de **X** para **S**. (A notação dos movimentos será explicado na Seção 2.2)



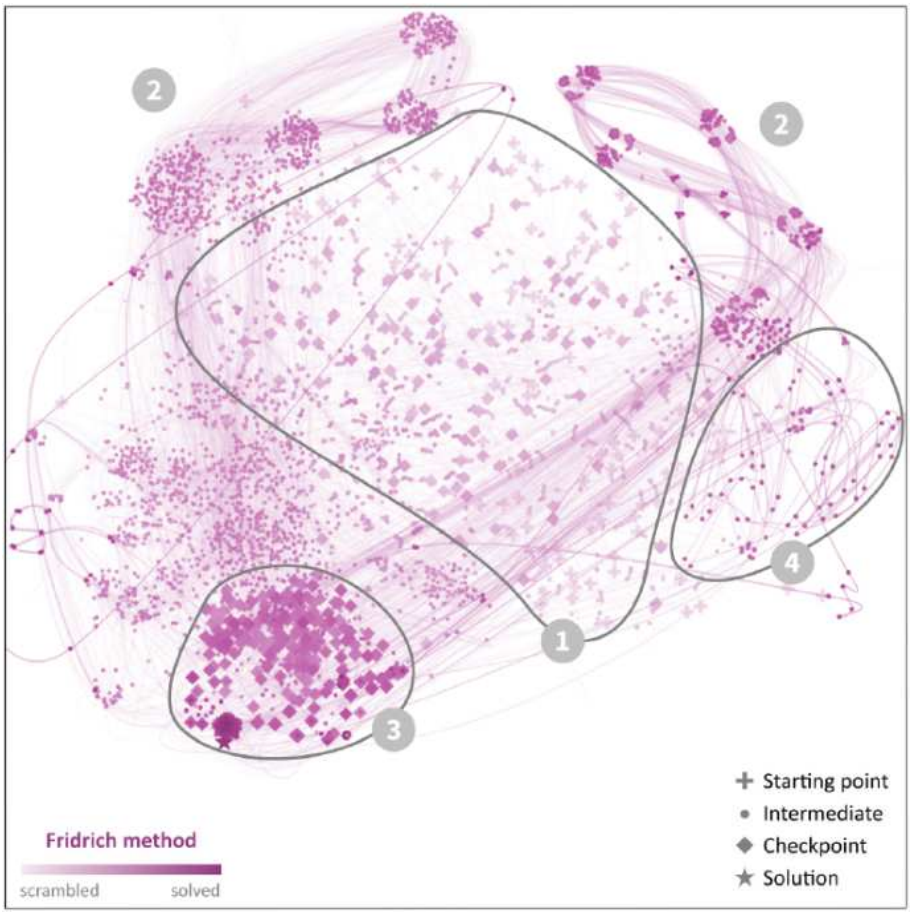
Fonte: Própria

1.2.2 Aprimorar o conceito de distância entre estados

Existem diferentes formas de medir a distância entre estados do cubo. No trabalho de visualização (STEINPARZ et al., 2019), por exemplo, utilizou-se as distâncias euclidianas entre os vetores que representam os estados do cubo (abordaremos modelagem na Seção 2.1). Porém, essa abordagem não reflete a real quantidade de movimentos necessários para levar de um estado a outro.

Com a abordagem proposta podemos definir o conceito de distância de forma mais apropriada à natureza do cubo. Com isso, é possível mapear os estados do cubo, criar grafos, analisar ciclos, clusterizar os estados e visualizar desempenho dos diversos algoritmos de solução de cubo (STEINPARZ et al., 2019).

Figura 2 – Projeção em 2D utilizando t-SNE de diversos casos resolvido pelo método de Fridrich.



Fonte: (STEINPARZ et al., 2019)

1.3 TRABALHOS ANTERIORES

Visualização: Em 2019 pesquisadores da Áustria publicaram um trabalho sobre a visualização dos estados do cubo a fim de analisar os métodos humanos de resolução de

cubo. O trabalho propõe um método de representação em 2D dos estados do cubo. Para isso, os autores utilizaram uma representação dos estados com 54 elementos onde cada elementos representa uma cor com um *one-hot encoding* que possui 6 valores binários, o que resulta em um vetor com 324 valores para cada estado. Para reduzir de 324 para 2 dimensões, os autores utilizaram a técnica T-distributed Stochastic Neighbor Embedding (t-SNE), que garante que estados com distâncias próximas fiquem em regiões próximas (STEINPARZ et al., 2019).

Para o trabalho, os autores utilizaram a distância euclidiana entre as representações dos estados, que na prática mede quantos adesivos diferentes há entre os estados, porém não mede a quantidade de movimentos de um estado para o outro (STEINPARZ et al., 2019).

Diâmetro do espaço de estados do cubo: Desde a criação do cubo, começou-se uma busca para descobrir qual é o menor número de movimentos suficiente para levar todos os estados do cubo para o estado resolvido. Posteriormente, esse número foi chamado de "God's Number".

Em 2010 um grupo de pesquisadores provaram que o God's Number é exatamente 20. O estudo considera a contagem por *half-turn metric* na qual rotações de 180^{o} são contadas como apenas um movimento. A outra forma de contagem é *quarter-turn metric* na qual rotações de 180^{o} são contabilizadas como duas rotações de 90^{o}.

O limite inferior do God's Number já havia sido provado ser 20 em 1995 por Michael Reid ao provar que um estado em especial precisava de exatos 20 movimentos para ser alcançado. Porém, em 2010, ao reduzir por simetrias os $4,3.10^{19}$ estados em cerca de $5,5.10^{7}$ junto do processamento de 35 CPU ano, o estudo provou que o limite superior também é de 20 movimentos. Além disso, também foi estimada a quantidade de estados distintos que são atingidos de acordo com a quantidade de movimentos, gerando a Tabela 1 (ROKICKI et al., 2014) que indica que a grande maioria dos estados do cubo estão a 17 ou 18 movimentos de distância do cubo resolvido.

Solução do cubo por programação dinâmica: No trabalho de (KORF, 1997), foi apresentada a proposta de utilizar uma variação da busca A* para atingir a solução ótima do cubo mágico, ou seja, a solução com a menor quantidade de movimentos possíveis. Para criar a heurística, o estudo propõe executar uma busca em largura para calcular todas as soluções ótimas de um subconjunto de peças, e guardá-las em tabelas. Exemplo: guardar todas as soluções ótimas para resolver qualquer estado das 8 quinas (modelagem do cubo será abordado na Seção 2.1). Depois de calcular diversas tabelas para diversos subconjuntos, a heurística se baseia em buscar o estado atual nessas tabelas, e retornar o máximo entre as soluções ótimas de cada subgrupo, provando assim, ser é uma heurística admissível.

O ganho de performance em comparação com uma simples busca em profundidade é expressivo, porém o método demanda muita memória e a pré computação das tabelas

Tabela 1 – *Canonical sequences* é a quantidade de estados na árvore de estados com d de profundidade. E *positions* são estados únicos para cada profundidade d.

d	Canonical sequences	Positions
0	1	1
1	18	18
2	243	243
3	3,240	3,240
4	43,254	43,239
5	577,368	574,908
6	7,706,988	7,618,438
7	102,876,480	100,803,036
8	1,373,243,544	1,332,343,288
9	18,330,699,168	17,596,479,795
10	244,686,773,808	232,248,063,316
11	3,266,193,870,720	3,063,288,809,012
12	43,598,688,377,184	40,374,425,656,248
13	581,975,750,199,168	531,653,418,284,628
14	7,768,485,393,179,328	6,989,320,578,825,358
15	103,697,388,221,736,960	91,365,146,187,124,313
16	1,384,201,395,738,071,424	≈1,100,000,000,000,000,000
17	18,476,969,736,848,122,368	≈12,000,000,000,000,000,000
18	246,639,261,965,462,754,048	≈29,000,000,000,000,000,000
19	3,292,256,598,848,819,251,200	≈1,500,000,000,000,000,000
20	43,946,585,901,564,160,587,264	≈300,000,000

Fonte: (ROKICKI et al., 2014)

tem um custo computacional elevado mesmo que esse custo seja diluído na resolução de diversos cubos.

2 METODOLOGIA

2.1 MODELAGEM DO ESTADO DO CUBO

Apesar de já termos usado o expressão "estado" previamente de forma intuitiva, segue a definição formal:

Definição 1 *O estado resolvido do cubo mágico é a configuração na qual cada uma de suas faces possui apenas uma das seis cores do cubo. Um estado do cubo é uma configuração única das cores de suas faces que pode ser atingida após efetuarmos uma sequência de movimentos partindo do estado resolvido.*

Para este trabalho, foram usados as três seguintes modelagens para representar um estado:

Representação por cores: Nessa modelagem, cada estado é representado por um vetor de 54 posições indexadas de 0 a 53, em que cada posição do vetor armazena o valor da cor de um determinado adesivo do cubo mágico. Para relacionar cada posição a um adesivo, utilizamos a indexação da Figura 3. E para ligar cada cor a um valor numérico, utilizamos o seguinte mapa de cor: 0 = Branco, 1 = Amarelo, 2 = Verde, 3 = Azul, 4 = Laranja, 5 = Vermelho. Como exemplo, a representação por cores do estado resolvido é o seguinte vetor:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5]

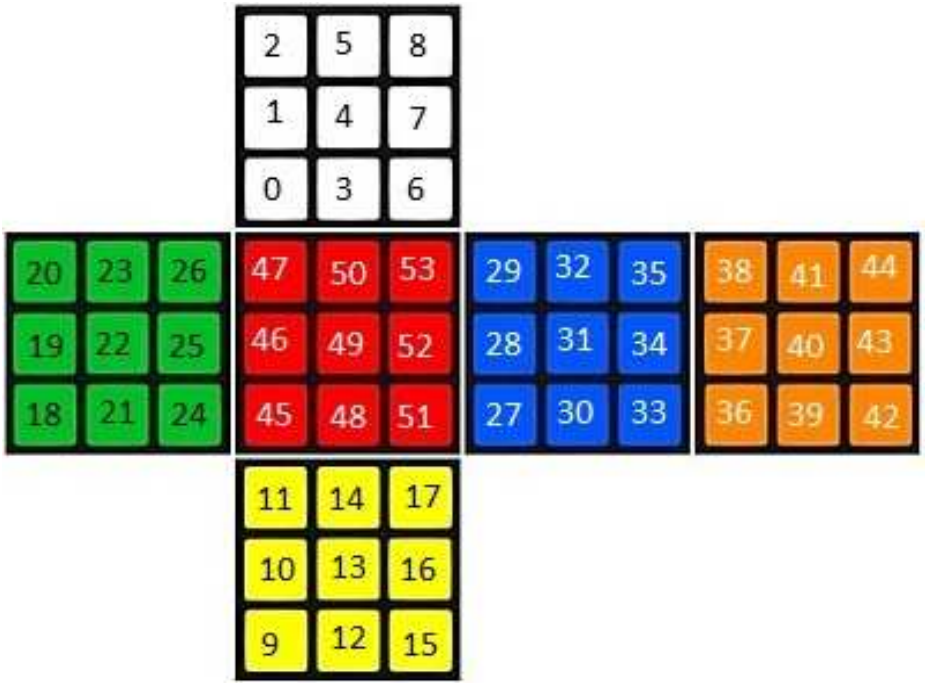
Representação com identificadores únicos: Nessa modelagem, cada estado é representado por um vetor de 54 posições indexadas de 0 a 53, em que cada posição do vetor armazena valor do identificador de um determinado adesivo do cubo mágico. Para relacionar cada posição a um adesivo utilizamos a indexação da Figura 3. E o identificador de cada adesivo é o valor da sua posição no cubo resolvido. Como exemplo, a representação com identificadores únicos do estado resolvido é o seguinte vetor:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]

A mudança da representação por identificadores únicos para a representação por cores se dá em apenas dividir o valor do identificador por 9 e pegar a parte inteira. Já o caminho inverso demandaria um processamento que analisa as cores vizinhas para retomar o identificador único de cada adesivo.

Representação por peças: Nessa modelagem, entendemos que o cubo é formado por peças. Uma peça é um conjunto de adesivos que estão sempre unidos independentemente

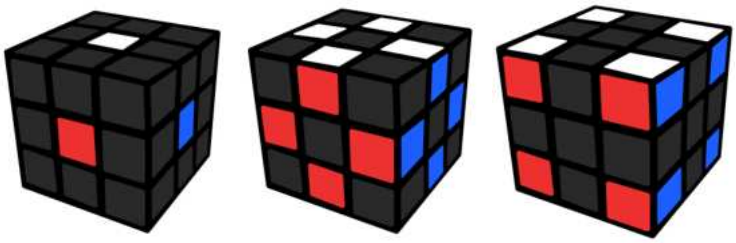
Figura 3 – Indexação dos adesivos utilizados neste trabalho.



Fonte: Própria

dos movimentos realizados, e por isso, podem ser tratados como um objeto só. O cubo possui três tipos de peças: peças de centro (Center Pieces) que possuem apenas uma cor visível; peças de meios ou bordas (Edge Pieces) que possuem duas cores visíveis; peças de quinas (Corner Pieces) que possuem três cores visíveis (vide Figura 4). Para a presente modelagem, serão representadas apenas as 12 peças de meios e 8 peças de quinas.

Figura 4 – Tipo de peças



Peças de centro, meios e quinas respectivamente.

Fonte: Própria

Cada peça possui 3 atributos:

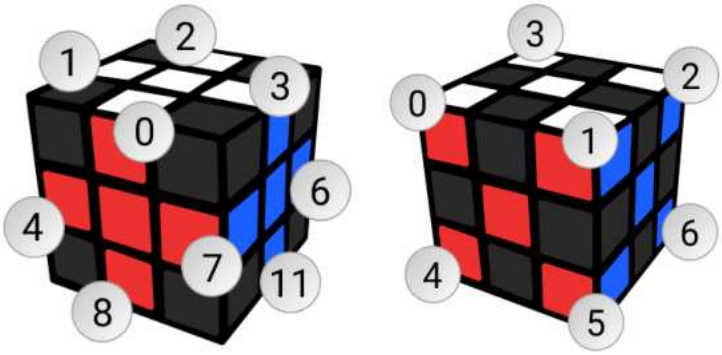
- **Posição:** É o espaço do cubo em que a peça se encontra. Para ligarmos cada espaço do cubo a um número, utilizamos a indexação sugerida pelo site CUBO VELOCIDADES

(Figura 5). Como uma peça de meio não pode ocupar o espaço de uma peça de quina e vice-versa, podemos indexar a posição dos 2 tipos de peça de forma independente.

- **Orientação:** As orientações de uma peça são as diferentes formas em que a peça pode se apresentar em uma determinada posição do cubo. Por exemplo, na Figura 6 ilustramos as 3 orientações possíveis para uma peça de quina em uma determinada posição. Para relacionar cada orientação a um número, será seguido um conjunto de regras apresentadas a seguir.

- **Identificação:** Cada peça é identificada por um número. Este número é determinado pela posição da peça no estado resolvido. Por exemplo, na Figura 5 a peça de quina que possui as cores vermelho-branco-azul é identificada pelo número 1 pôs no cubo resolvido ela se encontra na posição 1.

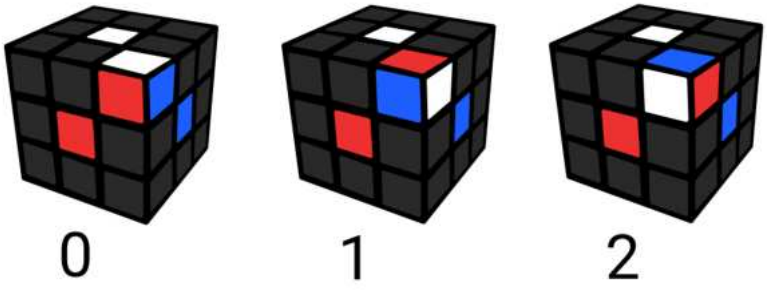
Figura 5 – Indexação das posições das peças



Note que temos um total de 20 posições, mas na figura apenas 16 estão representadas. As 4 restantes do fundo são denotadas por 7 (quina) e 5, 9 e 10 (dos meios). Fonte: (CERPE, 2007)

Fonte: Própria

Figura 6 – Orientações possíveis para uma quina

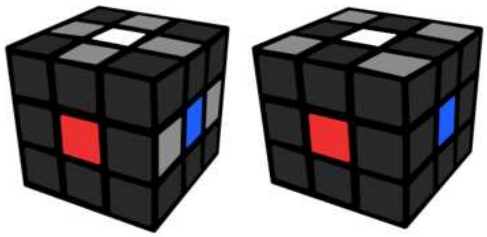


Fonte: Própria

Para representar orientações de forma numérica, utilizamos as seguintes regras: no caso das peças de quina, uma peça terá (i) orientação 0 se o adesivo branco ou amarelo estiver voltado a face branca ou amarela (sendo as cores das faces aquelas correspondentes ao estado resolvido do cubo); (ii) orientação 1 se a peça tiver rotacionada no sentido horário em relação a sua orientação 0; (iii) orientação 2 se tiver rotacionada no sentido anti-horário em relação a sua orientação 0 (vide exemplo na Figura 6).

No caso das peças de meios, uma peça terá (i) orientação 0 se tiver as cores branco ou amarelo voltadas para uma das regiões cinza da Figura 7 ou, caso a peça não tenha nem branco nem amarelo, terá orientação 0 se tiver as cores azul ou verde voltadas para uma das regiões já citadas; e (ii) terá orientação 1 caso contrário.

Figura 7 – Adesivos de referência de meios e de quinas.



Fonte: (CERPE, 2007)

Com essa modelagem de orientação podemos entender o valor da orientação de uma peça de quina como sendo quantas vezes a peça está girada a 120° sentido horário do seu estado orientado (com o valor da orientação igual a zero). Já no caso dos meios, representa quantas vezes a peça foi girada em 180° do seu estado orientado (com o valor da orientação igual a zero).

Nessa modelagem do cubo, para cada estado, utilizamos um vetor de tamanho 8 (resp., 12) para representar a posição das peças de quinas (resp., meios). Onde o i-ésimo elemento deste vetor contém o identificador da peça que encontra-se na posição i, e.g., se a peça 6 estiver na posição de quina 0, o vetor de posição das peças de quina conterá o valor 6 na posição 0. Para representar a orientação das quinas (resp., meios), utilizamos um vetor de tamanho 8 (resp., 12) onde o i-ésimo elemento deste vetor contém a representação da orientação (conforme as regras explicadas acima) da peça que ocupa a posição i do cubo. Nessa modelagem o estado resolvido é:

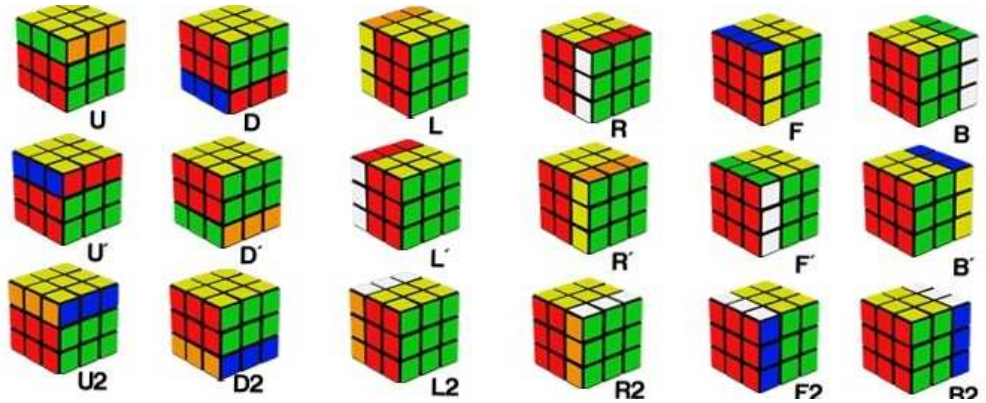
$$cornerPosition = [0, 1, 2, 3, 4, 5, 6, 7] \tag{2.1}$$
$$cornerOrientation = [0, 0, 0, 0, 0, 0, 0, 0] \tag{2.2}$$
$$edgePosition = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] \tag{2.3}$$
$$edgeOrientation = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \tag{2.4}$$
$$\tag{2.5}$$

Uso das diferentes representações: Cada uma das representações foram adotadas em diferentes partes deste trabalho. A representação por cores foi usada na entrada da rede neural artificial (ANN), a representação por peças foi usada para gerar o estado auxiliar (que será explicado na Seção 2.4.3.2) e a representação por identificadores únicos foi usada em todo o restante do processamento.

2.2 MODELAGEM DOS MOVIMENTOS E CAMINHOS

Para este trabalho usaremos a notação proposta por (SINGMASTER, 1981) no capítulo 3. O cubo possui 6 faces que podem ser rotacionadas em sentido horário e anti-horário. Cada face é representada por uma letra: F = frente, R = direita, L = esquerda, U = topo, B = costas, D = baixo. Um giro de 90° no sentido horário é representado pela a letra da face (Exemplo: [F, R, L, U, B, D]), um giro 90° no sentido anti-horário é representado pela letra da face seguindo de aspas simples (Exemplo: [F', R', L', U', B', D']) e para simplificar, dois giros seguidos na mesma face são representados pela a letra da face seguida do número 2 (Exemplo: [F2, R2, L2, U2, B2, D2]).

Figura 8 – Ilustração com a aplicação de cada movimento partindo do estado resolvido.



Fonte: (CINOTO, 2020)

Não há padrões que relacionem as cores as faces, porém, para todo este trabalho usamos a relação de cores e faces como: F = Vermelho, R = Azul, L = Verde, U = Branco, B = Laranja, D = Amarelo.

Também no escopo deste trabalho utilizaremos o termo caminho e a ação de aplicar um caminho seguindo a definição abaixo:

Definição 2 *Caminho é uma sequência de movimentos. A ação de aplicar um caminho em um estado determinado significa efetuar a sequência de movimentos a partir do estado no qual o caminho será aplicado.*

Por diversos momentos podemos nos referenciar a caminhos nomeando-os por letras como: Y = [R , F], então Y representa o movimento R seguido do movimento F. Nesse caso, a notação Y^2 representa o caminho [R, F, R, F] assim como [R, F]^2 também representa o mesmo caminho. Já a notação Y^{-1} representa o caminho [F', R'].

2.2.1 Transformações

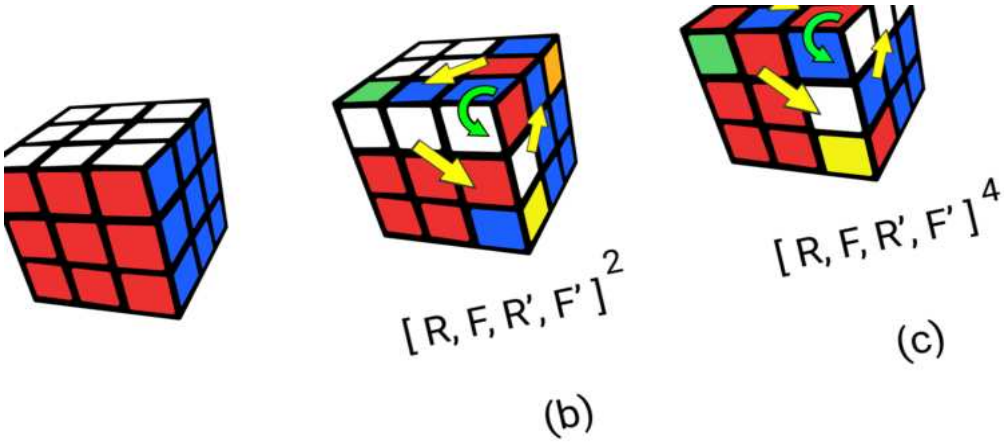
Todos os estados do cubo possuem o mesmo conjunto de peças, portanto, é natural entendermos que na diferença entre estados se resume na diferença entre a posição e a orientação de cada peça.

Quando aplicamos um caminho a um estado, é comum que o resultado final seja um estado diferente, então, podemos entender que um caminho efetua uma transformação na posição e na orientação das peças do estado de origem. Damos o nome de permutação ou rearranjo para o ato de alterar as posições das peças e ao ato de alterar as orientações damos o nome de rotação. Assim, podemos dizer que um caminho, quando aplicado, rearranja e rotaciona as peças de um estado do cubo.

Tanto o rearranjo quanto a rotação são completamente independentes do estado do cubo, ou seja, se tivermos uma transformação Y que troca a peça da posição 7 para a posição 3, mesmo que o estado tenha na posição 7 uma peça identificada por 4 ou 6, ao final da transformação Y a peça que ocupava a posição 7 será posicionada na posição 3. Podemos observar esse exemplo na figura 9 na qual a sequência de movimentos [R, F, R', F'] é aplicada 2 vezes ao estado resolvido (a) e assim atingindo o estado (b). Podemos notar que a peça de meio com as cores vermelho-azul que ocupa a posição 7 no estado (a) foi para a posição 3 no estado (b) da figura (a movimentação foi representada pela seta amarela sobre a face azul). Importante notar que ao aplicar a mesma sequência de movimento mais 2 vezes partindo do cubo (b) da figura e chegando ao cubo (c), notamos que a peça vermelho-branco que estava ocupando a posição 7 no estado (b) também foi posicionada na posição 3 no estado (c).

No caso da rotação, a alteração também é independente do estado do cubo, porém, o valor final da orientação da peça depende de sua orientação inicial. A rotação é de fato um giro na peça, portanto, saber qual será a orientação final de cada peça, depende de qual orientação a peça possuía inicialmente. Podemos entender melhor observando a peça de quina vermelho-branco-azul nos 3 estados da figura 9. Note que ao aplicar a sequência

Figura 9 – Ilustração da mesma sequência de movimentos sendo aplicada em diferentes estados e gerando o mesmo efeito.



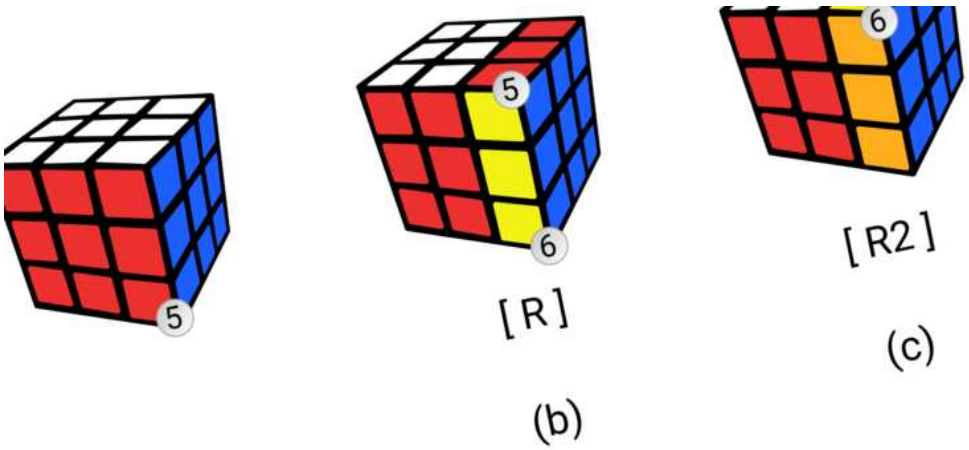
Fonte: Própria

de movimentos de (a) para (b) a quina que estamos observando saiu da orientação 0 para a orientação 2 (segundo o critério apresentado na modelagem de Representação por peças da Seção 2.1). Em seguida, note que aplicando a mesma sequência de movimentos no cubo (b) até chegar no cubo (c) a mesma peça partiu da orientação de valor 2 e terminou com orientação igual 1. Para modelar esse comportamento, podemos entender que a sequência de movimentos [R, F, R', F']^2 gira a peça de quina que o ocupa a posição 1 em 240° em sentido horário, portanto, o cálculo da orientação final é: orientação inicial vezes 120° mais 240° adicionados pela sequência de movimentos. No caso da orientação inicial ser 2, o valor da orientação final seria 480° porém isso é equivalente a 120° que na nossa modelagem corresponde ao valor 1. Vale notar que tanto os meios quanto as quinas só podem ser girados de forma discreta de 120° em 120° para as quinas e de 180° em 180° para os meios, portanto, as rotações possíveis só podem ser múltiplos desses valores, e assim, para simplificar, podemos utilizar diretamente o valor 1 quando a rotação for 120° e 2 em vez de 240°, no caso das quinas, e 1 para rotações de 180° para os meios. E para evitar valores que representem mais de uma volta inteira (360°) ao final da soma da orientação final podemos extrair o módulo 3 no caso das quinas, e módulo 2 no caso dos meios. Então, de forma geral: a orientação final de uma peça de quina é o valor da orientação inicial mais a rotação do movimento aplicado módulo 3 (ou módulo 2 no caso dos meios).

No exemplo acima mostrado, a peça que foi rotacionada tinha a mesma posição como origem e destino, mas e nos casos em que as peças têm posições de origem e destino diferente, precisamos considerar a orientação inicial da peça que ocupa a posição destino ou a posição origem? Para responder essa pergunta vamos observar o exemplo da figura 10. Na ilustração temos a peça identificada pelo número 5 na posição 5 do cubo (a) e sua

orientação é 0, em seguida executamos o movimento [R] para atingir o cubo (b) e então, podemos notar que a peça identificada por 5 agora ocupa a posição 1 e sua orientação é igual a 2. Observe também que a peça identificada por 6 no cubo (b) se encontra na posição 5, assim como a peça 5 originalmente, e possui a orientação igual a 1. Aplicando o mesmo movimento [R] no cubo (b) e chegando ao cubo (c), note que a peça identificada por 6 agora ocupa a posição 1, e sua orientação é igual a 0. Esse exemplo nos mostra que a orientação final de uma peça leva em consideração a orientação da peça na posição origem. Isso não é possível ser observado na transição de (a) para (b) pois, em relação a peça identificada por 5, tanto a orientação da peça que ocupa a posição 5 (origem) é 0, quanto a orientação da peça que ocupa a posição 1 (destino) também é 0. Porém, essa transição nos mostra que a peça 5 saiu da orientação 0 e chegou na orientação 2, então o movimento [R] rotaciona a peça que sai da posição 5 e vai para a posição 1 em 2 vezes (ou 240°). Agora, observando a transição de (b) para (c), em relação a peça 6, a orientação na posição 5 (origem) é igual a 1 e a orientação na posição 1 (destino) é igual a 2, no entanto, a orientação final da peça 6 no cubo (c) é 0, portanto a orientação final foi: orientação na peça na posição origem mais rotação da sequência de movimentos.

Figura 10 – Ilustração da mesma sequência de movimentos rotacionado e permutando peças simultaneamente.



Fonte: Própria

2.3 APRESENTAÇÃO DA DEEPCUBEA

A já citada DeepCubeA é uma abordagem que busca a solução do cubo através de uma busca A* ponderada, porém, com a função de heurística sendo implementada por uma *Artificial Neural Network* (ANN).

A busca A* procura o menor caminho entre um estado inicial e um estado final, explorando sempre o estado intermediário de menor custo. Sendo o custo calculado por $f(x) = g(x) + h(x)$, onde $g(x)$ é o custo entre o estado inicial até o estado x, e $h(x)$ é

uma heurística que estima o custo do estado x até o estado final. Para a DeepCubeA, foi usado uma variação da busca A* chamada *Weighted* A* que utiliza um peso para a função $g(x)$ portanto $f(x) = \lambda \cdot g(x) + h(x)$. Além disso, a implementação calcula um grupo de estados por vez, e com isso os pesquisadores nomearam o algoritmo de busca como '*batch weighted A* search*' (BWAS).

Para criar a função $h(x)$, foi desenvolvida uma ANN treinada por aprendizado profundo por reforço. A estratégia para treinar a rede é: partindo de um cubo resolvido, efetuar K movimentos aleatórios, entregar o estado resultante para a ANN e então bonificar a rede quando o valor retornado for o mais próximo de K. No começo são usados valores baixos para K e à medida que a rede vai acertando o valor de K aumenta até o limite de 30 movimentos.

A rede é formada por 2 camadas completamente conectadas onde a primeira camada contém 5.000 nós a segunda contém 1.000 nós. Em seguida há 4 blocos residuais cada um com 2 camadas com 1.000 nós cada. A saída consiste um único valor que é a $h(x)$. (AGOSTINELLI et al., 2019)

Por ser tratar de uma rede neural, não há como provar a admissibilidade da heurística, então o DeepCubeA não tem como garantir caminhos ótimos, segundo os casos de teste dos pesquisadores que desenvolveram a rede, o algoritmo retornou o caminho ótimo em 60% dos casos.(AGOSTINELLI et al., 2019)

2.4 PROBLEMA GERAL DO CUBO MAGICO

Nesta seção, descreveremos a solução do problema do cubo mágico. Iniciamos com duas definições.

Definição 3 *A solução tradicional do cubo mágico consiste num caminho de um estado conhecido A para o estado final no qual cada face do cubo tem apenas uma cor.*

Definição 4 *A solução geral do cubo mágico consiste num caminho de um estado conhecido A para outro estado conhecido B.*

2.4.1 Abordagem ingênua.

A abordagem ingênua para resolver o problema de múltiplas fontes para múltiplos destinos consiste em resolver duas vezes o problema de múltiplas fontes para único destino. Ou seja, resolvendo duas vezes o problema do cubo mágico tradicional, podemos resolver o problema geral do cubo mágico. Entretanto, tal solução apresenta alguns problemas:

1. **alto número de movimentos do cubo**: o número de operações será igual a soma do número de operações para ir do estado inicial para o estado final do cubo padrão, e de tal estado para o estado final desejado. É possível encontrar soluções muito melhores adotando-se estratégias alternativas.

2. **incapacidade de melhorar soluções existentes**: A proposta ingênua não permite melhorar uma solução não ótima. Nossa abordagem, em contrapartida, tem a vantagem de poder ser usada para refinar caminhos já preexistentes.

3. **custo computacional**: a abordagem ingênua tem custo computacional igual ao dobro do custo da solução tradicional. Já a solução alternativa terá um custo tipicamente menor.

2.4.2 Abordagem por ANN.

A DeepCubeA, depois de treinada, resolve o problema tradicional em 60% das vezes de forma ótima sem consumo relevante de memória e em tempo relativamente curto (AGOSTINELLI et al., 2019). Porém, para resolver o problema geral seguindo a mesma estratégia, precisaríamos de uma ANN que recebesse como entrada tanto o estado de origem quanto o estado de destino, e retornasse a estimativa do tamanho do caminho ótimo entre os estados. Essa estratégia apresenta os seguintes problemas:

1. **Alteração na ANN**: Seria necessário fazer alterações na rede neural pelo menos para receber como entrada os estados inicial e final. E possivelmente seria necessário ampliar arquitetura da rede no que diz respeito a quantidade de neurônios por camada e/ou quantidades de camadas. Já na nossa abordagem podemos aproveitar o DeepCubeA já treinado sem fazer qualquer alteração.

2. **Custo de retreinar**: A rede neural foi treinada para estimar a distância entre um estado qualquer e um único estado específico, que é o estado resolvido. Para estimar a distância entre quaisquer dois estados, seria necessário retreinar a rede. Para criar os casos de treino, seguindo a mesma lógica apresentada na seção 2.3, precisaríamos aplicar k movimentos partindo de um estado aleatório, e então dar como entrada na rede tanto o estado aleatório quanto o estado aleatório rotacionado k vezes, e bonificar a rede quando o valor retornado for próximo de k. Não há como estimar com precisão quanto recurso será necessário para retreinar a rede. Já a nossa abordagem utiliza a rede já treinada sem precisar fazer qualquer novo treinamento.

2.4.3 Nossa abordagem: estados auxiliares e caminhos equivalentes

Utilizando simetrias entre cubos podemos criar um estado auxiliar para cada caso do problema geral de tal forma que a solução tradicional do estado auxiliar é a mesma solução do caso geral. Assim resolvemos o problema geral sem precisar retreinar a rede ou fazer qualquer alteração no algoritmo que resolve o problema tradicional.

2.4.3.1 Independência entre estado e caminho

Todos os movimentos possíveis podem ser aplicados em qualquer estado do cubo (vide Seção 2.2). Assim, não há restrições entre estados e movimentos e portanto, todo e qualquer caminho pode ser aplicado a todo e qualquer estado. Dado um estado qualquer **A** e outro estado qualquer **B** achar um caminho **P** que leve de **A** para **B** se resume a achar um caminho que rearranje e rotacione as peças de tal forma que quando aplicado em **A** resulte em **B**. Porém, não precisamos efetivamente procurar **P** partindo de **A** e aplicar movimentos até chegar no estado **B**.

Podemos achar o caminho **P** aplicando movimentos a partir de qualquer estado inicial **X** até chegar a um estado **S** tal que o rearranjo e a rotação de **X** para **S** seja o mesmo rearranjo e a rotação necessários para levar de **A** para **B**. Considerando o estado **S** como sendo o estado resolvido do problema tradicional, podemos usar qualquer método que resolva o problema tradicional para descobrir o caminho **P** desejado.

Para demonstrar que o mesmo caminho **P** transforma de maneira equivalente dois estados diferentes temos: sendo **A** e **B** dois estados quaisquer do cubo, **X** o estado auxiliar baseado nesses dois estado e tomando a operação $\mathbf{B} - \mathbf{A}$ como sendo a transformação necessária para levar do estado **A** para o estado **B**, então, por construção, $\mathbf{S} - \mathbf{X} = \mathbf{B} - \mathbf{A}$ como demonstraremos na Seção 2.4.3.2. Sendo **P** a série de movimentos $(p_1, p_2, p_3, ..., p_n)$ que leva de **X** para o estado resolvido do problema tradicional **S** e tomando $\mathbf{X}_i$ como o estado **X** após ser aplicado a série de movimento de p_1 até p_i, inclusive, temos:

Supondo por contradição que o **P** leva de **X** para **S** e leva de **A** para **C**, tal que **C** seja diferente de **B**. Então, para algum i, $\mathbf{S} - \mathbf{X}_{i-1} = \mathbf{B} - \mathbf{A}_{i-1}$, porém, $\mathbf{S} - \mathbf{X}_i \neq \mathbf{B} - \mathbf{A}_i$. Isso implica afirmar que a transformação do movimento p_i se dá diferentemente ao estado $\mathbf{X}_{i-1}$ e ao estado $\mathbf{A}_{i-1}$, o que é um absurdo pois todos os movimentos são aplicáveis da mesma maneira a todos os estados do cubo.

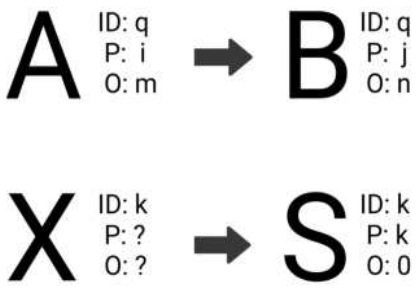
Na próxima seção mostraremos como construir um estado auxiliar **X** no qual o caminho de **X** para **S** rearranje e rotacione as peças da mesma forma que o caminho **P** que leva de **A** para **B**.

2.4.3.2 Estado auxiliar

O estado auxiliar **X** é um estado do cubo no qual, aplicando a solução do problema geral **P** (que leva de **A** para **B**) atingimos o estado resolvido do problema tradicional **S**. Para gerarmos esse estado, precisamos saber onde posicionar cada peça e com qual orientação.

No esquema da Figura 11 indicamos que a peça identificada por q se encontra na posição i com a orientação m no estado **A** e na posição j com orientação n no estado **B**. Quando temos um caso do problema geral, todas essas variáveis são conhecidas, também é conhecido que, no cubo resolvido, todas as peças se encontram na posição equivalente

Figura 11 – Esquema inicial da construção do estado X



Fonte: Própria

ao seu identificador e com orientação igual a zero, no entanto, o que não sabemos é como posicionar uma peça identificada por k no cubo **X** nem mesmo a relação entre k e q.

Para descobrir as relação acima citadas, utilizamos duas estratégias:

- **Posição:** Seja $p_A(q)$ a posição da peça de quina identificada por q no estado **A**, e seja $p_B(q)$ a posição final de tal peça no estado **B** após aplicar-se as transformações correspondentes ao caminho **P** em **A** (vide relação (2.6) abaixo). Seja $p_X(k)$ a posição da peça de quina identificada por k no estado **X**, e seja $p_S(k)$ a posição final de tal peça no estado resolvido **S** após aplicar-se **P** no estado **X** (vide relação (2.7) abaixo). Vale notar que todas as peças no estado resolvido **S** apresentam uma equivalência enter sua posição e seu identificador, então, $p_S(k) = k$ como exposto na Seção 2.1.

$$p_A(q) \xrightarrow{P} p_B(q) \tag{2.6}$$

$$p_X(k) \xrightarrow{P} p_S(k) = k \tag{2.7}$$

  Se o caminho **P** aplica a mesma transformação nos estados **A** e **X**, então, todas as peças que se originam na mesma posição tanto em **A** quanto em **X** devem ser destinadas para as mesmas posições finais em **B** e **S**, respectivamente. Em outras palavras, o lado esquerdo de (2.6) deve ser igual ao lado esquerdo de (2.7) e simultaneamente o lado direito de (2.6) deve ser igual ao lado direito de (2.7). Sendo assim:

$$p_A(q) = p_X(k) \tag{2.8}$$

$$p_B(q) = p_S(k) = k \tag{2.9}$$

$$p_B(q) = k \tag{2.10}$$

$$p_X(p_B(q)) = p_A(q) \tag{2.11}$$

  Logo, para construirmos o vetor de posições do estado **X** devemos respeitar a equação (2.11) para obter as propriedades desejadas. O argumento foi feito sobre o vetor

de posições das quinas, porém o mesmo argumento se aplica ao vetor de posições dos meios.

- **Orientação:** Seja $O_A(q)$ e $O_B(q)$ as orientações das peças de quina identificadas por q do estado **A** e do estado **B**, respectivamente, e $O_X(k)$ e $O_S(k)$ as orientações das peças de quina identificadas k do estado **X** e do estado **S**, respectivamente, temos:

$$O_A(q) \xrightarrow{P} O_B(q) \tag{2.12}$$
$$O_X(k) \xrightarrow{P} O_S(k) = 0 \tag{2.13}$$

Vale notar que todas as peças de **S** possui a orientação igual a 0 (zero) assim como apresentado na Seção 2.1. Para que **P** transforme as orientações dos estados **A** e **X** de forma equivalente, então, para toda peça que tem a mesma origem e o mesmo destino, deve sofrer a mesma rotação θ . Seguindo as relação expressa na equação (2.10) e o cálculo da orientação final apresentada na Seção 2.2.1 temos:

$$O_B(q) = O_A(q) + \theta \mod 3 \tag{2.14}$$
$$O_S(k) = O_X(k) + \theta \mod 3 \tag{2.15}$$
$$(O_B(q) - O_A(q)) \mod 3 = (O_S(k) - O_X(k)) \mod 3 \tag{2.16}$$
$$O_X(k) = (O_A(q) - O_B(q)) \mod 3 \tag{2.17}$$
$$O_X(p_B(q)) = (O_A(q) - O_B(q)) \mod 3 \tag{2.18}$$

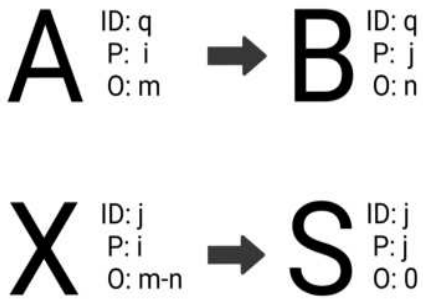
Então, para construirmos o vetor de orientação do estado **X** precisamos respeitar a equação (2.18) para obter as propriedades desejadas. O argumento foi feito sobre a orientação das quinas, porém, a orientação dos meios segue a mesma argumentação, com a alteração de $\mod 3$ para $\mod 2$.

Utilizando os nomes empregados no esquema da figura 11 temos que: $p_B(q) = j$ portanto, segundo a equação (2.10), $k = j$. Para a posição temos que $p_B(q) = j$ e $p_A(q) = i$ então, segundo a equação (2.11), $p_X(j) = i$. Para a orientação temos que $O_A(q) = m$ e $O_B(q) = n$ portanto, segundo a equação (2.18), $O_X(j) = m - n$. Assim temos o esquema completo expresso na figura 12.

2.4.3.3 O algoritmo

O algoritmo de criação do estado **X** recebe como entrada as representações por peças (vide Seção 2.1) do estados origem e destino, **A** e **B**, respectivamente, e retorna a repre-

Figura 12 – Esquema final da construção do estado X



Fonte: Própria

sentação por peças do estado **X** . A implementação desse algoritmo em linguagem Python está listada no Código 2.1.[1]

[1] O código fonte está também disponível no site do projeto: <https://github.com/GustavoMonteiroUFRJ/Rubik_Cube_Routing>

Código 2.1 – Cria estado auxiliar

```
1  import numpy as np
2
3  def createX(Acp, Aep, Aco, Aeo, Bcp, Bep, Bco, Beo):
4
5      # Declarando as estruturas de dados do estado auxiliar
6      Xcp = np.zeros_like(Acp)
7      Xep = np.zeros_like(Aep)
8      Xco = np.zeros_like(Aco)
9      Xeo = np.zeros_like(Aeo)
10
11     # Pre computando o index das quinas.
12     Bcp_index = np.zeros_like(Bcp)
13     for i in range(len(Bcp)):
14         Bcp_index[Bcp[i]]=i
15
16     # Criando o vetor de posicao das quinas do estado auxiliar
17     for i in range(len(Acp)):
18         Xcp[i] = Bcp_index[Acp[i]]
19
20     # Pre computando o index dos meios.
21     Bep_index = np.zeros_like(Bep)
22     for i in range(len(Bep)):
23         Bep_index[Bep[i]]=i
24
25     # Criando o vetor de posicao dos meios do estado auxiliar
26     for i in range(len(Aep)):
27         Xep[i] = Bep_index[Aep[i]]
28
29     # Criando o vetor de orientacao das quinas do estado auxiliar
30     for i in range(len(Aco)):
31         possicao_final = Bcp_index[Acp[i]]
32
33         orientacao_inicial = Aco[i]
34         orientacao_final = Bco[possicao_final]
35         Xco[i] = (orientacao_inicial - orientacao_final) %3
36
37     # Criando o vetor de orientacao dos meios do estado auxiliar
38     for i in range(len(Aeo)):
39         possicao_final = Bep_index[Aep[i]]
40
41         orientacao_inicial = Aeo[i]
42         orientacao_final = Beo[possicao_final]
43         Xeo[i] = (orientacao_inicial - orientacao_final) %2
44
45     return (Xcp, Xep, Xco, Xeo)
```

A seguir, descrevemos o código linha a linha.

Declaração de variáveis: As linhas 6-9 declaram os vetores da representação por peças do estado **X** (vide Seção 2.1). A função *zeros_like*() da biblioteca *numpy* recebe como parâmetro um vetor e retorna um vetor como o mesmo tamanho que o passado por parâmetro com todas as entradas zeradas.

Pré-calculando a posição das peças: Os loops das linhas 13-14 e 22-23 preenchem

as estruturas Bcp_index e Bep_index, respectivamente, que são vetores das posições das peças indexadas pelo identificador de peça. Ou seja, dado um identificador de peça, o Bcp_index retorna sua posição no vetor Bcp. Tais estruturas são utilizadas para otimização do código evitando múltiplas buscas nos vetores Bcp e Bep.

Preenchendo o vetor de posição das quinas e meios: O loop das linhas 17-18 preenche o vetor de posição das peças de quina conforme a equação (2.11) onde cada elemento da equação é representado no código da seguinte maneira: $p_A(q) = i$, $q = Acp[i]$, $p_B(q) = Bcp\_index[Acp[i]]$, $p_X(p_B(q)) = Xcp\_index[Bcp\_index[Acp[i]]]$. Com a linha 18 igualando $Xcp[i] = Bcp\_index[Acp[i]]$ implica na igualdade de $Xcp\_index[Bcp\_index[Acp[i]]] = i$ que substituíndo pelos elementos da equação temos $p_X(p_B(q)) = p_A(q)$ que é perfeitamente a equação (2.11). De forma totalmente equivalente, o loop das linhas 26-27 aplica a mesma estratégia para as peças de meios.

Preenchendo os vetores de orientação das quinas e dos meios: O loop das linhas 30-35 preenche o vetor de orientação das peças de quina conforme a equação (2.18) onde cada elemento da equação é representado no código da seguinte maneira: $O_B(q) = Bco[Bcp\_index[Acp[i]]]$, $O_A(q) = Aco[i]$, $O_X(k) = Xco[i]$. Fazendo as devidas substituições, a linha 35 equivale perfeitamente à equação (2.18). De forma semelhante, o loop das linhas 38-43 preenche o vetor de orientação dos meios apenas trocando de mod 3 para mod 2.

2.5 TEORIA DE GRUPOS

O conjunto de estados do cubo pode ser entendido como um grupo e iremos demonstrar isso a seguir. Com isso, uma pergunta natural é: o que a criação do estado auxiliar **X** representa em termos de teoria de grupo? Essa é a pergunta que iremos responder nesta seção além de expressar os problemas tradicional e geral do cubo mágico em linguagem de teoria de grupos.

Recapitulando o conceito de grupo: Grupo é uma estrutura utilizada na álgebra abstrata. Para um conjunto de elementos ser considerado como um grupo é preciso haver:

- **Operação fechada no grupo:** se x e y pertencem ao grupo G, então $x * y$ também pertence a G.
- **Elemento identidade**: sendo e o elemento neutro pertencente ao grupo G e x um elemento qualquer do grupo G então $x * e = e * x = x$.
- **Inversa para todo elemento do grupo:** para x pertence a G então existe x^{-1} também pertencente a G tal que $x * x^{-1} = e$.
- **Associatividade:** para todo x, y e z pertencentes a G, então, $x * (y * z) = (x * y) * z$

O grupo de estados do cubo mágico: Como expresso na definição 1, cada estado do cubo é uma configuração gerada a partir de movimentos aplicados no estado resolvido, então, todo estado por ser representado pela sequência de movimentos que leva do estado resolvido até o próprio estado. Assim, o conjunto de estados é formado por elementos que possuem a forma de: $[a_1, a_2, ..., a_n]$ onde todo a_i é um movimento do cubo, mais o estado resolvido é o elemento sem movimentos [ ] . Vale ressaltar que diversas sequências de movimentos pode resultar no mesmo estado, exemplo: [R, R'] = [R, U2, R', R, U2, R'] = [R', F, R, F']^6 = [ ]. Nos casos de sequências de movimentos que representam o mesmo estado, qualquer uma das sequências representa o mesmo elemento do conjunto. O conjunto de estado modelados dessa maneira representa um grupo pôs:

- **Operação fechada no grupo:** A operação '×' entre estados consiste em concatenar a sequência de movimentos, então:

  $[a_1, a_2, ..., a_n] \times [b_1, b_2, ..., b_n] = [a_1, a_2, ..., a_n, b_1, b_2, ..., b_n]$. Pelo resultado da operação ser tratar de uma sequência de movimentos, o resultado também é um elemento do grupo.

- **Elemento identidade**: O elemento neutro é o próprio estado resolvido dado que $[a_1, a_2, ..., a_n] \times [] = [a_1, a_2, ..., a_n] = [] \times [a_1, a_2, ..., a_n]$.

- **Inversa para todo elemento do grupo:** E a inversa é a sequência invertida $[a_1, a_2, ..., a_n] \times [a_n^{-1}, a_{(n-1)}^{-1}, ..., a_1^{-1}] = []$.

- **Associatividade:** $[a_1, a_2, ..., a_n] \times ([b_1, b_2, ..., b_n] \times [c_1, c_2, ..., c_n]) = ([a_1, a_2, ..., a_n] \times [b_1, b_2, ..., b_n]) \times [c1, c2, ..., cn]$

Com essa modelagem por grupo, podemos reescrever o problema tradicional do cubo como sendo :

$$\hat{A} \times T = \hat{S} \tag{2.19}$$

Onde $\hat{A}$ é o elemento do conjunto que representa o estado que desejamos resolver, T é a resposta, e $\hat{S}$ é o elemento do conjunto que representa o estado resolvido. Como $\hat{S}$ é a identidade do grupo de estados do cubo mágico, então $T = \hat{A}^{-1}$. Então o desafio de de resolver o cubo é descobrir a inversa do estado do cubo. Já o problema geral do cubo mágico pode ser escrito da seguinte maneira.

$$\hat{A} \times P = \hat{B} \tag{2.20}$$

Onde $\hat{A}$ é o elemento do conjunto que representa o estado original, $\hat{B}$ é o elemento do conjunto que representa o estado destino e P é a solução do problema. Utilizando as operações algébricas de grupo temos:

$$\hat{A} \times P = \hat{B} \tag{2.21}$$
$$\hat{A}^{-1} \times \hat{A} \times P = \hat{A}^{-1} \times \hat{B} \tag{2.22}$$
$$[] \times P = \hat{A}^{-1} \times \hat{B} \tag{2.23}$$
$$P = \hat{A}^{-1} \times \hat{B} \tag{2.24}$$

Sendo o estado auxiliar X o estado em que aplicando a solução do problema geral obtemos o estado resolvido do problema tradicional e $\hat{X}$ o elemento do conjunto que representa esse estado, temos:

$$\hat{X} \times P = \hat{S} \tag{2.25}$$
$$\hat{X} \times \hat{A}^{-1} \times \hat{B} = \hat{S} \tag{2.26}$$
$$\hat{X} \times A^{-1} = B^{-1} \tag{2.27}$$
$$\hat{X} = \hat{B}^{-1} \times \hat{A} \tag{2.28}$$

Ou

$$\hat{A} \times P = \hat{B} \tag{2.29}$$
$$(\hat{B}^{-1} \times \hat{A}) \times P = \hat{S} \tag{2.30}$$
$$\hat{X} = \hat{B}^{-1} \times \hat{A} \tag{2.31}$$

Interpretando as equações: Como apresentado na equação 2.19, o problema original parece trivial já que é simples achar a inversa de uma sequência de movimentos, porém, quanto temos efetivamente um cubo embaralhado, não temos como saber diretamente qual foi a sequência de movimento que gerou esse estado, portanto, não temos como saber qual é a inversa desse estado trivialmente. Já no problema geral, temos a mesma característica, apesar de sabermos a configuração de cores do estado A (origem) e do estado B (destino), não sabemos efetivamente as sequências que levam de S para A nem de S para B.

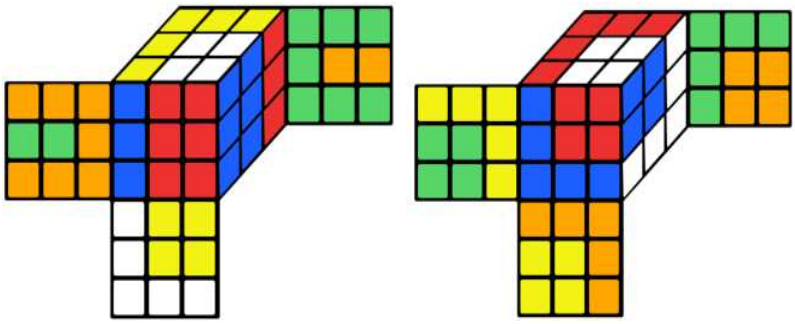
Outra observação, é que a equação 2.24 nos diz que P é a junção da solução do problema tradicional do estado A ($\hat{A}^{-1}$) e a inversa da solução do do problema tradicional do estado B ($\hat{B}$) que é exatamente a solução ingênua citada na seção 2.4.1. Seguindo a equação 2.31, podemos obter o estado X a partir do caminho de B para S concatenado com a inversa do caminho de A para S, o problema é que no geral não temos esses caminho, e justamente, a vantagem da nossa abordagem é conseguir gerar o estado X sem precisar achar as soluções tradicionais de A e B.

3 RESULTADOS

3.1 EXEMPLO

Tomando como exemplo dois padrões propostos no site (RUWIX, 2019), chamaremos de **A** o padrão "Displaced Motif" e de **B** o padrão "Cube in the cube". **A** é atingido quando, partindo do estado resolvido, aplicamos: L2 B2 D' B2 D L2 U R2 D R2 B U R' F2 R U' B' U' e **B** é atingido quando, partindo do estado resolvido, aplicamos F L F U' R U F2 L2 U' L' B D' B' L2 U. Ambos os estados estão visualmente expressos na Figura 13.

Figura 13 – Visualização dos estados **A** e **B** respectivamente.



Fonte: Própria

A representação por peças são:

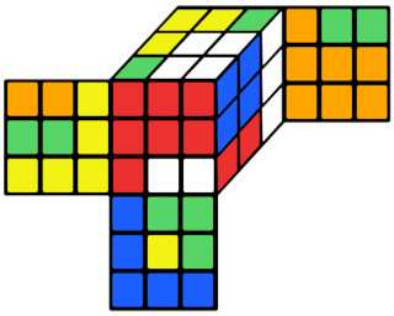
$$ACornerPosition = [6, 1, 4, 7, 2, 5, 0, 3] \tag{3.1}$$
$$ACornerOrientetion = [0, 10, 9, 3, 6, 5, 4, 7, 8, 2, 1, 11] \tag{3.2}$$
$$AEdgePosition = [0, 0, 0, 0, 0, 0, 0, 0] \tag{3.3}$$
$$AEdgeOrientetion = [0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0] \tag{3.4}$$

$$BCornerPosition = [5, 1, 0, 4, 6, 2, 3, 7] \tag{3.5}$$
$$BCornerOrientetion = [0, 8, 4, 3, 11, 5, 1, 7, 6, 9, 10, 2] \tag{3.6}$$
$$BEdgePosition = [2, 0, 2, 1, 1, 2, 1, 0] \tag{3.7}$$
$$BEdgeOrientetion = [0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1] \tag{3.8}$$
$$\tag{3.9}$$

Aplicando o algoritmo para criar o estado auxiliar X temos:

$$XCornerPosition = [4, 1, 3, 7, 5, 0, 2, 6] \tag{3.10}$$
$$XCornerOrientetion = [0, 10, 9, 3, 8, 5, 2, 7, 1, 11, 6, 4] \tag{3.11}$$
$$XEdgePosition = [2, 0, 2, 0, 1, 1, 1, 2] \tag{3.12}$$
$$XEdgeOrientetion = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0] \tag{3.13}$$

Figura 14 – visualização do estado auxiliar **X**



Fonte: Própria

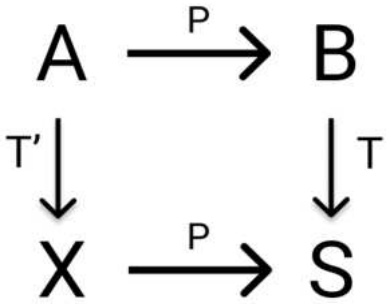
Utilizando a DeepcubeA para resolver o estado X temos um caminho P com 15 movimentos em contagem quarter-turn, que levou 5.122068s e visitou 16957 estados:

P = [L, B, D', B, R, R, F', R', D, F, D, R', B', L', D']

3.1.1 Desigualdade das relações de B com S e A com X

A construção do estado auxiliar X foi criada observando apenas as transformações de A para B, porém, buscamos entender se poderia haver outra forma de se atingir o estado X.

Figura 15 – Diagrama completo.



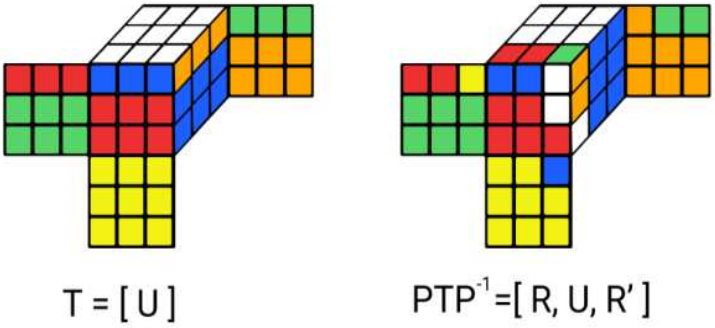
Fonte: Própria

Segundo o diagrama da figura 15, T é a transformação de B para S e T' é a transformação de A para X. Caso T e T' sejam iguais, não precisaríamos olhar para as transformações

entre A e B para criar X, poderíamos avaliar apenas as transformações de B para S, e então aplicar essas transformações em A para obter X.

No entanto, afirmar que T = T' implica em dizer que T = PTP^{-1} o que não é sempre verdade e pode ser trivialmente demonstrado na figura 16 com o caso de T = [U] e P = [R].

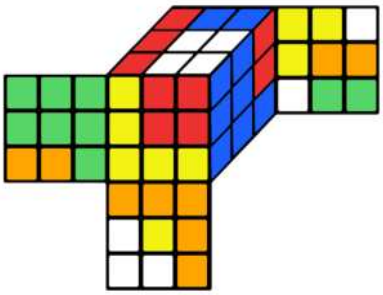
Figura 16 – Exemplo em que T não equivale a PTP^{-1}



Fonte: Própria

Como experimento, aplicamos a ideia no caso explorado no exemplo da seção 3.1. Nesse caso, a transformação T pode ser feita através dos movimentos inversos para se atingir B partindo de S (T = [U' L2 B D B' L U L2 F2 U' R' U F' L' F']), e assim, aplicando T em A atingimos o estado ilustrado na figura 17 que não é igual ao estado X ilustrado na figura 14.

Figura 17 – Estado A transformado por T



Fonte: Própria

3.2 MATRIZ DE DISTÂNCIAS E RESULTADOS NUMÉRICOS

Escolhemos como exemplo 5 padrões listados no site (RUWIX, 2019) sendo eles: "gift box", "cube in the cube", "displaced motif", "six spots", "order in chaos" (denominamos [E_1,E_2,E_3,E_4,E_5]). Aplicando a mesma abordagem para medir as distâncias 2 a 2, geramos 25 estados auxiliares para serem resolvidos e assim medir a distância de E_i para E_j. Dos

25, 5 representavam a distância de E_i para E_j para $i = j$, sendo assim, o estado auxiliar era o próprio estado resolvido, e sua solução tem tamanho zero. Também havia redundâncias já que o caminho ótimo de E_i para E_j terá o mesmo tamanho do caminho ótimo de E_j para E_i. Porém, aplicamos a solução em todos esses casos para validar a DeepCubeA.

Os resultados depois de resolver os 25 casos se expressam pela seguinte matriz, onde o valor na posição i, j é o tamanho do caminho retornado pela DeepCubeA para o caminho que leva de E_i para E_j.

$$\begin{pmatrix} 0 & 28 & 25 & 24 & 28 \\ 30 & 0 & 19 & 20 & 26 \\ 27 & 15 & 0 & 27 & 33 \\ 28 & 24 & 25 & 0 & 24 \\ 30 & 26 & 27 & 30 & 0 \end{pmatrix} \tag{3.14}$$

A máquina utilizada foi uma g4dn.xlarge da AWS que possui uma GPU NVIDIA T4 que vem integrada com 2.560 CUDA Cores e 320 Tensor Cores (NVIDIA, 2019). Tirando os 5 triviais, o tempo médio de execução para cada resolução foi de 14,68449s. A quantidade média de estados visitados por solução foi de 49136 e o tamanho médio das soluções foi de 25,8 movimentos.

Pela matriz apresentada é possível ver que os caminhos redundantes que deveriam ter o mesmo tamanho apresentaram tamanhos diferentes, o que é uma prova de que o DeepCubeA de fato não retorna apenas os caminhos ótimos.

Utilizando apenas a melhor opção entre os caminhos redundantes, temos a seguinte matriz:

$$\begin{pmatrix} 0 & 28 & 25 & 24 & 28 \\ 28 & 0 & 15 & 20 & 26 \\ 25 & 15 & 0 & 25 & 27 \\ 24 & 20 & 25 & 0 & 24 \\ 28 & 26 & 27 & 24 & 0 \end{pmatrix} \tag{3.15}$$

Tirando os caminhos de tamanho zero, o tamanho médio dos caminhos nessa matriz com as melhores soluções é de 24,2 movimentos. Interessante notar que as distâncias não divergem muito da média, o que vai na direção da ideia de que os estados são aproximadamente equidistantes uns dos outros.

3.2.1 Desempenho do algoritmo de criação de estado auxiliar

Para medir o desempenho do algoritmo de criação do estado auxiliar, foi feita uma amostra com 400 estados aleatórios, e foi criado o estado auxiliar para cada uma das combinações 2 a 2 dos estados. O experimento foi executado em um computador com

processador Intel Core I5-3550 de 3,30 GHz, o algoritmo foi escrito em python 3.7 executado na plataforma Jupyter Notebook. Foram geradas 79.800 execuções que levaram o total de 393,75094223 segundos, uma média de 0,0049342 segundos por execução. Valor cerca de 0,0336% do tempo médio de solução do cubo.

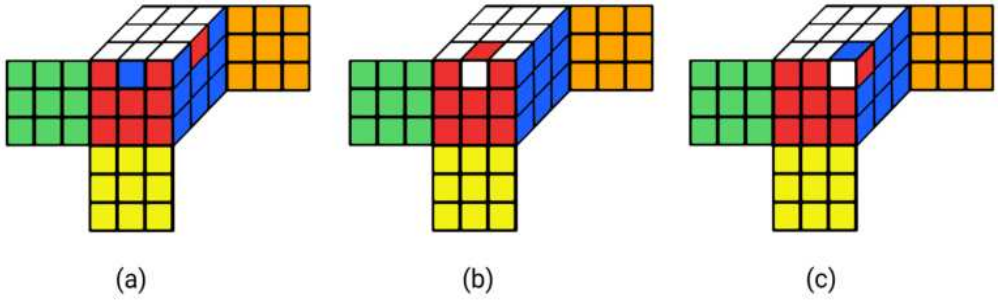
3.3 REFINO DE SOLUÇÕES SUB-ÓTIMAS

Com o experimento descrito na seção anterior foi descoberto inesperadamente que a generalização do problema do cubo mágico poderia ajudar a refinar a solução do problema tradicional do cubo mágico. O princípio se baseia no fato de que todo caminho do cubo é reversível, e se o caminho P leva o estado resolvido S para um estado desejado A, então o caminho P^{-1} leva o cubo de A para S. Nos algoritmos que não resolvem o cubo de forma ótima, é possível que ao resolver o problema de A para S o caminho resultante seja diferente do caminho inverso do problema de S para A. E assim, usando a técnica de criação do cubo auxiliar é possível resolver o problema de S para A e comparar o resultado com o caminho de A para S e então escolher o caminho mais curto.

3.4 EXPANDINDO O MÉTODO PARA OUTROS ESPAÇOS

Ao longo de todo o trabalho foi utilizado apenas o espaço de estados do cubo mágico clássico. Porém existem diversos quebra-cabeças com espaços de estados idênticos ao do cubo mágico. Utilizando a própria implementação física do cubo mágico é possível forçar configurações que não são possíveis de retornar ao estado resolvido utilizando apenas os movimentos do cubo mágico. Podemos observar algumas dessas configurações na figura 18. No entanto, essas novas configurações geram um espaço de estados perfeitamente iguais ao espaço de estados do cubo mágico.

Figura 18 – Casos impossíveis



Fonte: Própria

Chamando os espaços gerados por configurações impossíveis de componentes conexas alternativas, é simples demonstrar que todo elemento de uma componente conexa pode ser relacionado a um estado do cubo mágico de forma única. Cada componente conexa é fruto

de uma ação que não pertence ao conjunto de movimentos do cubo mágico, porém essa ação pode ser revertido em qualquer estado da componente conexa e assim retornando ao espaço de estados do cubo.

Porém, supondo que temos duas configurações, A e B, e sabemos que pertencem a mesma componente conexa mas não sabemos qual, no entanto, queremos achar o caminho que leva de A para B. Se A e B são alcançáveis entre si então existe um caminho P que contém apenas movimentos do cubo mágico que quando aplicado em A resulta em B. Supondo que se saiba qual é o caminho P, é possível aplicar P^{-1} em S e obter X. Logo, para cada caso geral de uma componente conexa fora do espaço de estados do cubo mágico, existe um X associado dentro do espaço de estado do cubo mágico. Se X fosse externo ao espaço de estados do cubo, implicaria dizer que P^{-1} quando aplicado em S gera um estado fora do espaço de estados do cubo o que é contraditório com a própria definição de estado do cubo mágico.

4 CONSIDERAÇÕES FINAIS

Há décadas o cubo mágico e outros quebra-cabeças tridimensionais têm atraído a atenção dos pesquisadores. Este trabalho trouxe ao cubo mágico a ideia de resolver o problema geral a partir da solução de um sub conjunto mais restrito do mesmo problema, o que pode influenciar futuras pesquisas a aplicar a mesma ideia em outros quebra cabeças e até explorar as limitações que essa ideia pode ter.

Graças a perfeita simetria do espaço de estados do cubo, foi possível desenvolver a técnica de estado auxiliar que modela qualquer caminho no espaço de estados do cubo para um problema tradicional do cubo mágico. Para provar o conceito, utilizamos o DeepCubeA por ser um algoritmo que faz uso de inteligência artificial para resolver o cubo mágico de forma relativamente rápida, eficiente e principalmente com baixo custo computacional. O principal algoritmo desenvolvido foi o que gera o estado auxiliar para ser enviado ao DeepCubeA, mas também foi preciso desenvolver a estrutura de dados da representação por peças (necessária na criação do estado auxiliar) e a transformação dessa representação por peças para a representação por identificadores únicas utilizada pelo DeepCubeA. Com essas implementações, podemos ver que a criação dos estados auxiliares não trouxe *overhead* significativo, permitindo assim resolver o problema geral do cubo mágico com custo computacional equivalente ao custo de resolver o problema tradicional.

Garantia de caminhos ótimos. A DeepCubeA se demonstrou rápida e com resultados melhores do que algoritmos humanos de solução de cubo, porém sobre caminhos ótimos deixou dúvidas já que 50% dos caminhos retornados (excluindo os triviais) estavam acima do número limite de 26 movimentos quarter-turn. De qualquer maneira, para este trabalho, o algoritmo cumpriu a função de resolver o problema tradicional do cubo. Caso seja necessário obter o caminho ótimo entre 2 estados quaisquer do cubo, então, se faz necessário o uso de métodos que utilizam programação dinâmica que garantem caminhos ótimos por utilizar heurísticas admissíveis. Porém para algum caso em que não seja possível usar tais métodos que garantem otimalidade, a abordagem apresentada neste trabalho trouxe a possibilidade de melhorar respostas de algoritmos não ótimos ao custo de resolver um caso do problema geral.

Visualização. Com a nova abordagem é possível refazer o trabalho de visualização com a t-SNE (STEINPARZ et al., 2019) utilizando o conceito de distância mais apropriado à natureza do cubo. Possivelmente os gráficos serão mais intuitivos para analisar os diferentes algoritmos de solução do cubo. O desafio se dá no processamento da distância de uma grande quantidade de estados já que para criar uma matriz de distâncias, é preciso calcular distâncias 2 a 2, o que cresce $O(n^2)$ sendo n a quantidade de estados. O trabalho original analisou 100 resoluções utilizando o método de Fridrich. E como o método em

média resolve o cubo em 56 movimentos (RUBIKSPLACE, 2011), a principal imagem no trabalho contém cerca de 5600 estados do cubo. Com a DeepCubeA utilizando o mesmo cenário do presente trabalho, gerar a matriz de distâncias necessária para refazer o trabalho original demandaria mais de 7 anos de processamento.

Teoria de grupos. A teoria de grupos é uma área do conhecimento que estuda o fenômeno da simetrias e o espaço de estados do cubo é perfeitamente simétrico, então se torna natural a relação que os dois assuntos possuem. Com a linguagem de teoria de grupos conseguimos descrever a existência do estado auxiliar e como ele se relaciona com o estado origem e destino. Porém, foi a partir da observação da natureza do cubo, uma boa modelagem tanto do estados do cubo quanto das transformações que as sequências de movimentos geram nos estados do cubo que foi possível montar o estado auxiliar sem precisar resolver o problema tradicional para os estados origem e destino. O sucesso dessa abordagem no caso do cubo mágico abre a dúvida se é possível expandir para outros problemas que também podem ser modelado como grupos e quais características esses grupos precisam ter para também obterem sucesso.

REFERÊNCIAS

AGOSTINELLI, F. et al. Solving the rubik's cube with deep reinforcement learning and search. **Nature Machine Intelligence**, v. 1, n. 1, p. 356–363, 2019. DOI:10.1038/s42256-019-0070-z.

CERPE, R. **CUBO 3X3X3 - MÉTODO BLINDFOLDED**. 2007. [Online; acessado 16-Novembro-2019]. Disponível em: <http://www.cubovelocidade.com.br/tutoriais/cubo-magico-blindfolded.html>.

CINOTO. **Notação dos movimentos**. 2020. [Online; acessado 04-junho-2020]. Disponível em: <http://cinoto.com.br/cubomagico/notacao-dos-movimentos/>.

KORF, R. E. Finding optimal solutions to rubik's cube using pattern databases. In: **AAAI/IAAI**. [S.l.: s.n.], 1997. p. 700–705.

NVIDIA. **Datasheet Nvidia t4 Tensor core GPU**. 2019. [Online; acessado 25-Novembro-2019]. Disponível em: <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-datasheet-951643.pdf>.

ROKICKI, T. et al. The diameter of the rubik's cube group is twenty. **SIAM Review**, SIAM, v. 56, n. 4, p. 645–670, 2014.

RUBIKSPLACE. **How to Speedsolve the Rubik's Cube**. 2011. [Online; acessado 16-Novembro-2019]. Disponível em: <http://www.rubiksplace.com/speedcubing/guide/>.

RUWIX. **Pretty Rubik´s Cube patterns with algorithms**. 2019. [Online; acessado 16-Novembro-2019]. Disponível em: <https://ruwix.com/the-rubiks-cube/rubiks-cube-patterns-algorithms/>.

SINGMASTER, D. **Notes on Rubik's magic cube**. [S.l.]: Enslow Publishers Hillside, NJ, 1981.

STEINPARZ, C. A. et al. Visualization of rubik's cube solution algorithms. The Eurographics Association, 2019.

GLOSSÁRIO

caminho Uma série de movimentos do cubo magico.

orientação Atributo de uma peça que indica qual das diferentes maneiras uma peça pode se apresentar em uma determinada posição. *veja também* posição

overhead Custo computacional não vinculado diretamente com a resolução do problema principal. No caso deste trabalho, é o custo computacional a mais para resolver o problema geral do cubo mágico em comparação com o custo para resolver o problema tradicional do cubo mágico.

solução geral Um caminho que leva o cubo mágico do estado presente ao estado resolvido (onde o estado resolvido pode ser qualquer estado pré-determinado). *veja também* caminho

solução tradicional Um caminho que leva o cubo mágico do estado presente ao estado resolvido padrão (todas os elementos de cada face com mesma cor). *veja também* caminho