Solving a Rubik's Cube Using Genetic Algorithm

ABHISHEK SARKAR* and ADARSH BHARADWAJ*, Birla Institute Of Technology And Science, India

The solution to a Rubik's Cube in a random state has been solved using Genetic Algorithm with a fitness function based on the number of incorrectly placed stickers in the cube.It is an evolutionary search algorithm where the solution is searched based on selection of the few of the closest states of the cube to the required solution in the current generation.These solutions are then combined and mutated to produce children with a better fitness value and the process is repeated until the required solution state is obtained.

CCS Concepts: • **Computing methodologies->Machine learning->Machine learning approaches->Bio inspired approaches->Genetic algorithm**;

Additional Key Words and Phrases: rubiks cube, genetic algorithm, 2d cube representation

ACM Reference Format:

Abhishek Sarkar and Adarsh Bharadwaj. 2023. Solving a Rubik's Cube Using Genetic Algorithm. In . ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Genetic algorithms are a computational method that are inspired by the principles of natural evolution and are used to find solutions to problems with a large search space without mentioning a clear set of rules for finding the optimal solution. They work by iteratively evolving a population of candidate solutions through a series of generations, using techniques such as selection, crossover, and mutation to create new solutions and improve the quality of the population over time.

The Rubik's cube is a classic puzzle that involves rearranging the colored squares on a cube to match a specific goal configuration. The search space for the Rubik's cube is vast, as there are many possible configurations that the cube can be in. This makes it an ideal problem for solving using a genetic algorithm.

To solve the Rubik's cube using a genetic algorithm, the configuration of the cube can be represented as a string of genes that encode the positions and colors of the squares. A fitness function can be used to evaluate the performance of each candidate solution (i.e., for each configuration of the cube) and guide the evolution of the population of solutions through a series of iterations. The genetic algorithm can search the space of possible solutions and find a sequence of moves that will transform the starting configuration of the cube into the goal configuration.

One of the key advantages of using a genetic algorithm to solve the Rubik's cube is the ability to adapt to changes in the problem through crossover and mutation. This allows the algorithm to converge to a good solutions even in the absence of a clear set of rules for finding a solution.

*Both authors contributed equally to this research.

Manuscript submitted to ACM

There are 43,252,003,274,489,856,000 possible states in a 3x3x3 Rubik's Cube. These makes it extremely challenging to obtain the required solution state both efficiently and with relatively lower computational power.Rubik's Cube has 6 centres, 6 faces or sides, 12 edges and 8 corners.All the pieces except the centres can be oriented.

The number of permutations can be calculated by the following:-

- Centres: Only 1 orientation
- Corners: There are a total of 8 corners and out of these,7 corners can be oriented independently and 3 states for each of the corner. The number of permutations will then be:-

$$8! * 3^7$$

- Edges: There are a total of 12 edges and the combination of centres,edges and corners are restricted to have an even permutation. [9]Moreover, each of the edge has a flip orientation.The number of permutations will be:-

$$12!/2 * 2^{11}$$

Thus,the total number of possible permutations will be the product of all these sub-permutations possible:-

$$8! * 3^7 * 12!/2 * 2^{11} = 43,252,003,274,489,856,000$$

In this paper,we would be using the standard Singmaster Notation [2] for representing the various moves in the cube. The notation is as follows:-

- F/F' : Front face rotated clockwise/anticlockwise
- U/U' : Upper face rotated clockwise/anticlockwise
- D/D' : Upper face rotated clockwise/anticlockwise
- L/L' : Left face rotated clockwise/anticlockwise
- R/R' : Right face rotated clockwise/anticlockwise
- M/M' : Middle layer rotated clockwise/anticlockwise with respect to right face (moves like E/E' and S/S' have been omitted as they are easily substituted by x/y/z and M/M' moves)
- B/B' : Back face rotated clockwise/anticlockwise
- x/y/z and x'/y'/z' : Rotating the entire cube about X-axis, Y-axis and Z-axis clockwise and anticlockwise respectively.

2 LITERATURE REVIEW

The earlier works in the field of finding optimal solutions to the Rubiks Cube was done by Morwen Thistlethwaite [8] in 1985 which was later improved by Herbert Kochiemba in 1992.[6]

There are two prominent and State-Of-The-Art algorithms present today to solve the Rubiks Cube:-

One of the algorithms uses Deep Reinforcement Learning (DRL) [1] [5] where the initial state space is defined as the initial random configuration of the cube and action space is the set of possible moves which can be taken by the cube.The agent will receive a positive reward if it is closer to the required solution and vice versa. Using this reward function, a policy is made to select the correct action until the goat state is reached. This is combined with a Monte Carlo Tree Search (MCTS) which is also popularly used in AlphaZero which is used for other combinatorial games and puzzles like chess, Go and shogi. This algorithm was called DeepCubeA and it had found the shortest path to the goal state about 60 percent of the time.

The other algorithm is the Korf's Algorithm [3], which is an Iterative Deepening A* algorithm (IDA*) that is used to find the optimal solution to a problem in the least amount of time. It is a variant of the popular A* algorithm, which is commonly used for path-finding problems and other search problems.The main idea behind Korf's algorithm is to use a heuristic function, which is the number of misplaced pieces to guide the search for a solution. The heuristic function is used to estimate the cost of reaching the goal state from the current state, and the algorithm uses this estimate to prioritize which states to expand next. The algorithm begins by setting a threshold value for the estimated cost, and then repeatedly expands states until a solution is found or the threshold is exceeded.

In summary, DRL and Korf's algorithm are both powerful techniques for solving the Rubik's cube, but they have their own unique advantages and disadvantages [4]. DRL is more flexible and can learn to optimize the heuristic function used by Korf's algorithm, but it can require a lot of computational resources and data to train effectively. Korf's algorithm is a well-established technique that is relatively simple to implement, but it can be slower to execute and less flexible than DRL. By flexibility, we are refering to the adaptability of the algorithm to changes in initial and goal configurations and the constraints. For instance, let us assume that the goal state we are trying to achieve is some arbitrary state of the cube. In this case, genetic algorithm and the DRL algorithm would adapt to the new constraints whereas the Korf algorithm may not.

3 THEORY

Before explaining how solving the Rubiks cube can be converted into a Genetic algorithm problem, the cube's various faces, orientations and moves must be converted into code which is then fed in the algorithm. We have created a Cube class, which would include a constructor for initializing all the faces of the cube to a 3x3 matrix corresponding to the solved state of the cube. This class will also include functions for every move of the cube which will orient these matrices accordingly and other utility functions for calculating fitness, obtaining the scramble, etc..

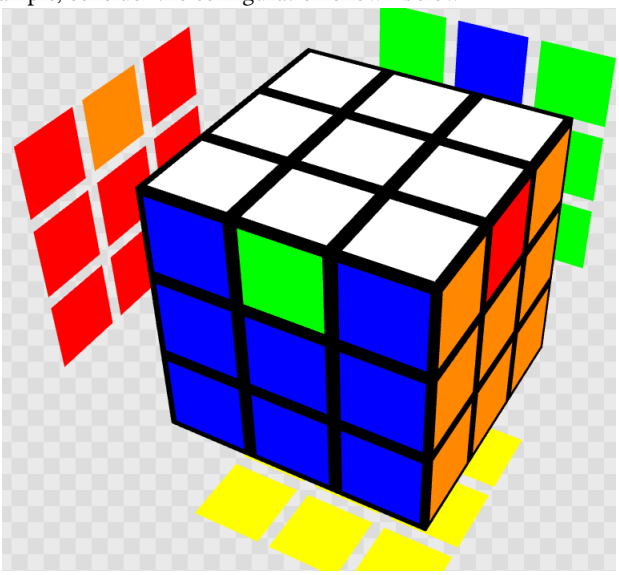
Genetic algorithms (GA) can be used to solve the Rubik's cube by using a population-based search strategy to find the optimal sequence of moves to solve the cube. The state space can be a representation of the cube's current configuration. In our study, it is represented as a 2D image of 6 3x3 matrices representing each face of the cube. The fitness function can be considered as the number of misplaced stickers in the cube. If the fitness function is closer to zero, this would mean that we are closer to the required solution. The population is initialized with a set of randomly generated solutions. Each solution/configuration of the cube is represented as a chromosome, which is a sequence of moves. The selection process is used to select the best solutions from the population to be used for crossover and mutation. The selection process can be based on the fitness function, with the best solutions having a higher probability of being selected. The crossover process is used to combine two solutions to create a new solution. The new solution inherits some of the characteristics of both parents. Crossover can be performed by randomly selecting a point in the chromosome and swapping the remaining part of the chromosome. The mutation process is used to introduce small random changes to a solution. These mutations help for a faster convergence to the required solution. It can be performed by randomly selecting a move in the chromosome and replacing it with a new move. The selection, crossover, and mutation steps are repeated until a solution is found or a stopping criterion is met.

The Genetic algorithm for solving the Rubiks cube stands as middle ground between the flexibility of the DRL algorithm and the computationally efficient Korf algorithm.

4 SIMULATIONS

Python was used for the simulation process as libraries like matplotlib and pycuber help with the visualization of the cube and numpy helps with manipulation of matrices. While running the algorithm, we have considered a population size of 500 in every generation. The population size must be sufficiently high to ensure a large search space for the algorithm for the finding the moves with better fitness scores. However,it should not be too large as the algorithm would waste a lot of time searching randomly without crossover or mutations. A maximum of 300 generations have been considered after which the training will reset as we assume convergence to a solution will not occur and a maximum of 10 such resets are considered, although a maximum of only 1 reset has occurred when our simulation was conducted. A total of 50 elitists with the best fitness scores are considered which will move on to the next generation.

To make the convergence faster, we have hard coded certain high probability configurations as set moves. For example, consider the configuration shown below:-

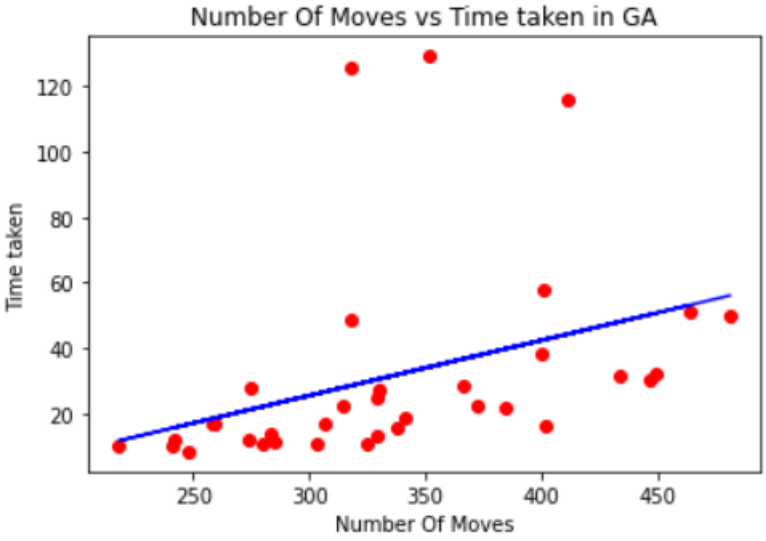


This is a high probability configuration when genetic algorithm is used. It might also be challenging to solve using our algorithm as improving the fitness function further cannot be accomplished by simple U moves. The set moves for this configuration are : M2 U M2 U2 M2 U M2. Similarly, multiple set moves for high probability configurations have been considered to solve the algorithm more efficiently.

5 RESULTS

A total of 35 observations were made in our simulation and our model of genetic algorithm was able to solve the rubiks cube at an average of about 30 seconds and at an average of 336 moves. Although the fitness function acts as a heuristic to guide the algorithm towards the solution, our algorithm uses random search in between generations before selecting the elitists which is the cause of the large number of moves. As an example,for the scramble : R' U' L2 B2 U2 F L2 B' L' B D R B F2 L F R' B2 F' L B' D B2 R2 D' U B2 F' D R2 , when the algorithm has been simulated multiple times,the time taken to solve the cube ranged from 8 seconds to 129 seconds.

We have observed a linear trend when plotting the number of moves against the time taken to solve the cube, which was expected.



6 CONCLUSION

Overall, Genetic algorithm is extremely useful to solve the Rubik's Cube with limited computational power. However, it is important to note the shortcomings of this algorithm. Fundamentally,the algorithm uses random search in each generation and tries to improve the fitness function which might make the algorithm quite inconsistent in terms of solving the cube, both in time taken and number of moves. Although solving the Rubiks Cube in God's Number [7] using genetic algorithm might be far-fetched, there is scope to obtain sub-optimal solutions using this algorithm. One of the promising approaches to solve this problem is by including set moves for more sophisticated configurations. Fine tuning the parameters might also help reduce the solving time.

REFERENCES

[1] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1(8):356–363, 2019.

[2] Lindsey Daniels. Group theory and the rubik's cube. *Lakehead Universityc*, 2014.

[3] Richard E Korf. Finding optimal solutions to rubik's cube using pattern databases. In *AAAI/IAAI*, pages 700–705, 1997.

[4] Zefeng Lyu, Zeyu Liu, Anahita Khojandi, and Andrew Junfang Yu. Q-learning and traditional methods on solving the pocket rubik's cube. *Computers & Industrial Engineering*, 171:108452, 2022.

[5] Stephen McAleer, Forest Agostinelli, AK Shmakov, and Pierre Baldi. Solving the rubik's cube with approximate policy iteration. In *International Conference on Learning Representations*, 2019.

[6] Tomas Rokicki. Twenty-five moves suffice for rubik's cube. *arXiv preprint arXiv:0803.3435*, 2008.

[7] Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. The diameter of the rubik's cube group is twenty. *siam REVIEW*, 56(4):645–670, 2014.

[8] Morwen Thistlethwaite. Thistlethwaite's 52-move algorithm, 1981.

[9] MICHAEL TRAVIS. The mathematics of the rubik's cube. *University of Chicago*, 2007.