**NEURAL RUBIK’S – SOLVING RUBIK’S CUBE USING NEURAL NETWORK (HEURISTIC LEARNING)**

**BY GUNASEGARRAN MAGADEVAN**

**PROPOSAL SUBMISSION FOR MASTER OF DATA SCIENCE**

**FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

**UNIVERSITY MALAYA**

**2019**

# CONTENTS

**PAGE**

[CONTENTS 2](#_Toc530249495)

[CHAPTER 1 3](#_Toc530249496)

[INTRODUCTION 3](#_Toc530249497)

[1.1 Research Background 3](#_Toc530249498)

[1.2 Motivation 4](#_Toc530249499)

[1.3 Research Problem 5](#_Toc530249500)

[1.4 Objective 5](#_Toc530249501)

[1.5 Research Question 5](#_Toc530249502)

[1.6 Research Delivery 6](#_Toc530249503)

[1.7 Research Signification 6](#_Toc530249504)

[CHAPTER 2 7](#_Toc530249505)

[LITERATURE RIVIEW 7](#_Toc530249506)

[2.0 Introduction 7](#_Toc530249507)

[2.1.1 Pattern Database Learning 9](#_Toc530249508)

[2.1.2 Baseline Learning 10](#_Toc530249509)

[2.1.3 Boosting Learning 11](#_Toc530249510)

[2.1.4 Heuristic Learning 11](#_Toc530249511)

[2.1.5 Comparation of the Learnings 13](#_Toc530249512)

[CHAPTER 3 14](#_Toc530249513)

[RESEARCH METHODOLOGY 14](#_Toc530249514)

[3.1 Quantitative Research 14](#_Toc530249515)

[3.2 Informed Search Algorithm 15](#_Toc530249516)

[3.2.1 Admissible Heuristics 15](#_Toc530249517)

[3.2.1.1 Max Misplaced Edge Cubes in Face 15](#_Toc530249518)

[3.2.1.2 Max Misplaced Corner Cubes in Face 16](#_Toc530249519)

[3.2.2 Inadmissible Heuristics 17](#_Toc530249520)

[3.2.1.3 Max Misplaced Corner and Edge Cubes in Face 17](#_Toc530249521)

[3.2.1.4 Max Plus Min Misplaced Corner and Edge Cubes in Face 17](#_Toc530249522)

[3.2.1.5 Max Misplaced Corner Cubes Plus Max Misplaced Edge Cubes in Face 17](#_Toc530249523)

[3.3 Train the model 18](#_Toc530249524)

[3.3.1 Approximate Function 18](#_Toc530249525)

[3.4 Compare the results 20](#_Toc530249526)

[REFERENCE 21](#_Toc530249527)

# CHAPTER 1

## INTRODUCTION

### Research Background

Neural networks have already proven to be able to cope with noisy and unstructured data like hand-written texts, images, sounds, classification of real-world objects based on incomplete description, and many others.

Recently, they also succeeded in several purely combinatorial domains like the game of AlphaGo. Nowadays, the program AlphaGo (D.Silver, 2016) that utilizes a deep neural net can beat top-class human players which was impossible just two years ago. No other approach is currently known to be able to play AlphaGo on such level. Neural Networks are also a key component of the best Poker engine DeepStack (Matej Moravcˇík, 2017) and several attempts have been made to use them for solving instances of Travelling Salesman Problem and other combinatorial problems (HieuPham, 2016).

In development, the machine learning approaches are already being used in several ways, for example to select the best search algorithm, preprocess the problem or to promote searching of promising areas.

Some attention has also been dedicated to heuristic learning, where the task is to automatically induce a heuristic function from training samples using a machine learning model. Models typically used in this area are very simple and are not finetuned for the specific problem. In most cases, they are only used as a black box.

With recent rapid development of deep learning models, many new possibilities are now available in this area. Learning algorithms now exist for efficient training of deep feed-forward networks and also many other types of Neural Networks have been developed and successfully used. For instance, there are Deep Recurrent Networks like Long Short – Term Memory (LSTM) (Francisco Javier Ordóñez, 2016), Deep Convolutional Networks, and Neural Turing Machines (Ariel Felner, 2016).

### Motivation

Rubik's Cube makes use of mathematical group theory, which has helped deduce certain algorithms. Furthermore, the fact that there are well-defined subgroups in the Rubik Cube group enables the puzzle to be learned and mastered by moving through different " difficulty levels " in itself. These subgroups are the principle underlying the computer cubing methods by Thistlethwaite and Kociemba, which solve the cube by further reducing it to another subgroup. (Boris Gorshenev, 2018)

There are now a number of solutions that can solve the cube in less than 100 steps. David Singmaster first published his solution in 1981, which solves the cube layer by layer. A team of researchers who worked with Google in July 2010 has proved that the so- called " number of God” (minimum number of moves to solve any) was 20. The Herbert Kociemba's Two-Phase Algorithm is the used for the most move optimal online Rubik's Cube solver programs, which typically calculates a solution of 20 steps or less.

Since all these types of solutions require very tough mathematics to fully understand why they work and needs tremendous amount of mathematical logics to develop new solutions, it is very difficult for ordinary people and even impossible for computers to develop these solutions automatically.

With the hope that computer systems will learn how to solve Rubik's Cube with some general algorithm and after carefully searching previous work online, finally found that the most popular methods that is used to solve Rubik's Cube problem are reinforcement learning. However, this technique requires so much computer power, therefore, finding alternatives. Since not found any neural network methods in heuristic learning, believe that this is because the objective function space is too large for neural networks to learn, but it's still worth attempting because there's no relevant experiment to prove it - and at least this method to solve Rubik's Cube with neural networks in heuristic learning could also fill the gap and provide data for this method.

### Research Problem

* Implementing heuristic learning to solve a given instance Rubik’s cube problem.
* Minimal number of predictions of moves to solve a given instance Rubik’s cube.

### Objective

* Using heuristic learning to construct a automatically solve a given instance Rubik’s cube problem.
* Train neural network to predict a minimal number of moves required to solve a given case of Rubik’s cube.
* Trained neural network as a heuristic distance estimator with a standard forward-search algorithm and compare the results with other heuristics.

### Research Question

* This research and experiment will show that the neural network, heuristic learning approach is competitive with state-of-the-art and might be the best choice in some use-case scenarios.

### Research Delivery

* To prove this research and experiment of neural network using heuristic learning is the best choice in some use-case scenarios to solve Rubik’s Cube.

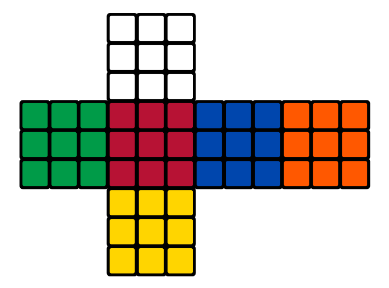
### Research Signification

* Train a neural network to predict a minimal number of moves required to solve a given example of Rubik's cube. At that point utilize the trained system as a heuristic distance estimator with a standard forward-search algorithm and compare the results with other heuristics.

# CHAPTER 2

## LITERATURE RIVIEW

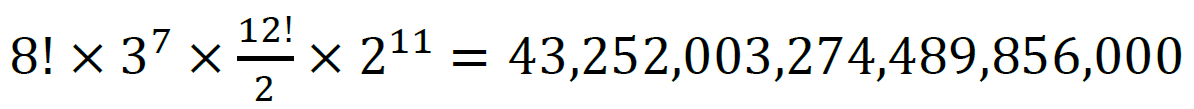
#### Introduction



**Figure 1.1: Rubik’s Cube in 2 – Dimensional View.**

Rubik’s cube, the famous puzzle, may serve as an example of a planning problem as its task is to find a sequence of actions leading form given initial state to specified goal state.

From a search perspective, the number of unique states of the standard 3x3x3 cube is ([(8! x 38) x (12! x 212)]/12), which are over 43 quintillion legal positions of the Rubik’s Cube.



**Figure 1.2 Rubik’s Cube Formula.**

At approximate 26 quarter-moves, every state can be solved optimally. There is a single goal state and the branching factor is 12 when using a symmetry breaking representation. We consider a model utilizing quarter-move meaning that it is permitted to turn layers just by 90◦. The half-move representation allows sides to be rotated by 180◦ in a single move.

To be able to describe some different properties of the cube, we first need to define a couple of extra thoughts. The 3 by 3 solid shape comprises of 27 little blocks which is known as cubies. Some faces of these cubies are colored, or more precisely they carry colored stickers. In the goal state, all faces of the large cube contains only stickers of the same color.

In the event that we dismantled the cube square, we would get 8 corner- cubies, each of which carrying 3 colored stickers, 12 edge- cubies, each with 2 stickers on them and 6 central-cubies each having 1 color.

A cube of any size can easily be solved sub optimally using a simple algorithm that runs in time Θ (n2), where n is the size of the cube. Finding optimal solutions seems to be substantially harder, in spite of the fact that the complexity class of this task has long been unknown. It has only recently demonstrated that unravelling Rubik’s cube optimally is certainly NP- hard (M. Rudoy, 2017). Current state-of-the-art approaches for finding ideal or close ideal frequently utilize forward state space search using a pattern database as a heuristic (Nathan R.Sturtevant, 2014).

##### Pattern Database Learning

Pattern databases (PDBs) are precomputed answers for littler issues which are made from the first by abstracting-away some of its features. For instance, in the event that I just consider the 8 corner cubies of a standard 3x3x3 cube, I will be getting a cube’s which is considerable easier to solve. The entire state space of this smaller issue might be listed, solved optimally for each state and stored in a database. For each state of the standard cube, its will be able for me extend it on the example by ignoring everything except the corners, look-up this state in the database and use its evaluation as a lower bound on the length of the plan. The set of features that were consider in the smaller problem is called pattern and features that that are excluded in the pattern are ignored.

A single pattern never contains all cubies. Utilizing it as a heuristic lead to a state where all cubies contained in the pattern are effectively set however others are most certainly not. In such states, the heuristic value is 0 and the algorithm experiences considerable difficulties finding the goal state since it has no further guidance. For efficient searching it is necessary to combine several PDBs with different patterns or to combine PDB with other types of heuristics. There are many ways of combining the heuristics, from simple ones, like taking maximum, to more complex ones like additive PDBs or cost partitioning (Florian Pommerening, 2016).

##### Baseline Learning

Baseline method, neural network architecture for the weak learner. All architectures tested were trained on the task of predicting the correct move sequence when given a sequence of cube states. To encode each cube state, I use a one-hot encoding of sticker colours. There are 54 colors, and each can be one of 6 colors. Indicator variable *X*i,j is 1 if the *i*th sticker is the *j*th color, and 0 otherwise. This gives a total of 54 × 6 = 324 features.

An alternative representation, I did not try was representing the cube by the position and orientation of the 8 corner cubes and 12 edge cubes. The sticker-based representation was easier to implement, and initial results were promising enough to continue further work. Nevertheless, a cube representation may be worth pursuing.

Every baseline architecture used ends with 12 outputs, corresponding to the 12 possible face turns - each of the 6 faces can be rotated clockwise or anti-clockwise.

Different architecture choices (fully connected and recurrent) were tested while keeping the number of parameters in the model constant. The best results were achieved with LSTMs. (Kociemba, Herbert, 2016)

##### Boosting Learning

Boosting algorithms are a general class of ensemble methods that convert several weak learners into a single strong one. They have theoretical guarantees on convergence rates for error, but occasionally fail in practice. (Bengio, Yoshua, 2009)

They start by reviewing boosting algorithms from the literature, before proposing and testing their own variant. For the rest of this paper, *f*t denotes the weak learner for the *t*th iteration, *D*t is the distribution over data that ft is trained on, and *F*t is the strong learner at the end of the *t*th iteration. They also use standard classification notation, where *X* is the data space and *Y* is the label space. (Alexander Irpan, 2016)

##### Heuristic Learning

The task of heuristic learning is to consequently create a heuristic function for given problem based on some training data. The learning is normally completed a priory, where the training data are provided before the search. Heuristic can likewise be learned on-the-fly, where previous attempts to solve the problem serve as training data to learn future search strategy (Springer-Verlag, 2012).

Several attempts have been done to utilize NNs for the heuristic learning task (Austin Dionne, 2011). In the typical setting, an arrangement of features is computed for each state in the training set as well as the optimal distance-to-go to the nearest goal state, and the network is then used to learn a mapping from features to distance estimate. After the learning procedure is done, the system is utilized as a heuristic distance estimator together with an informed forward search algorithm like A\* or IDA\*.

Heuristics that are learned in this way provide no guarantees on admissibility. The objective of learning is with the goal that the heuristic would be close to the real value but not necessarily always admissible. For example, smaller than the real distance-to-go. Since search with an inadmissible heuristic doesn’t typically guarantee finding optimal solutions, this approach is just reasonable in situations where close-to- optimal solutions are sufficient. It is however possible to guarantee optimality even with an inadmissible heuristic by modifying the search strategy (Erez Karpas and Carmel Domshlak, 2012).

From the complexity perspective, the task of heuristic learning in general is hard. It is known that the task of finding ideal answers to some planning problems like generalized 15-puzzle, Sokoban, and many more is NP-hard or even harder. It is also known that with an accurate-enough heuristic, the search time may be polynomial. To be specific, if ∀x : (h∗(x) − h(x)) ∈ O(log(h∗(x))) where x are states, h is heuristic and h∗ is the real goal distance, then the search time is polynomial (Stuart J. Russell and Peter Norvig, 2010). It is therefore obvious that such heuristic cannot be computed in polynomial time unless some complexity classes collapse. The learned heuristic will most likely not have the capability to solve large problems optimally in polynomial time, but it may still outperform classic human-designed heuristics.

##### Comparation of the Learnings

**Table 1.2: Comparation of the Learnings**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Learnings** | **Pattern Database** | **Baseline** | **Boosting** | **Heuristic** |
| **Difference** | A pattern is the partial specification of a permutation (or state); that is, the tiles occupying certain locations are unspecified. | A baseline prediction algorithm provides a set of predictions that you can evaluate as you would any predictions for the problem. | Boosting is a general method for enhancing the performance of a learning algorithm. It is a method for finding an exceptionally precise classifier on the preparation set, by joining "weak hypotheses”, every one of which needs just to be decently exact on the preparation set. | A heuristic is a general dependable guideline, normally created through understanding, that is joined into basic leadership process keeping in mind the end goal to tackle an issue. Numerous chart look calculations, for example, A∗, utilize heuristic-based pursuit. |

# CHAPTER 3

## RESEARCH METHODOLOGY

###### Quantitative Research

Quantitative research methods are research methods dealing with numbers and anything that is measurable in a systematic way of investigation of phenomena and their relationships. It is applied when a question is answered based on relationships within measurable variables with an intention to explain, predict and control a phenomenon (Leedy 1993).

An entire quantitative study usually ends with confirmation or disconfirmation of the hypothesis tested. Researchers using the quantitative method identify one or a few variables that they intend to use in their research work and proceed with data collection related to those variables.

Using a scientific approach, quantitative methods often deal with results computation and system analysis in the field of Information Technology (IT). The objective of quantitative method is to develop and employ models based on mathematical approach, hypotheses and theories pertaining to the nature of an IT phenomenon. The process of measurement is the focus on quantitative method due to its connectivity between empirical observation and mathematical expression of quantitative relationships, which is also known as iterative process where evidence is evaluated, and hypotheses and theories are refined with some technical advances, leveraging on statistical approach.

Data collection based on a hypothesis or theory and continues with application of descriptive or inferential statistics is where quantitative method typically begins. Surveys and observations are some examples that been used widely with statistical association.

For an example, when a researcher is interested to investigate the “*effectiveness of expert system for managing an application in open source environment*”, the researcher will formulate the research question such as, “*How effective is the expert system in comparison to case-based reasoning for an application module development*”.

###### Informed Search Algorithm

The strategies used in this project design is admissible heuristics with different heuristics, and machine-learned estimators.

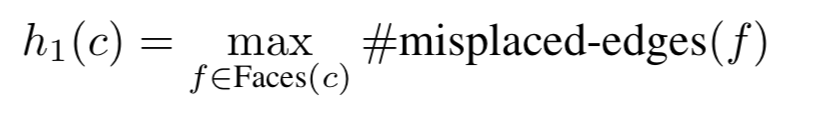
**Table 1.2: Strategies used in observation studies.**

|  |  |  |
| --- | --- | --- |
| **No.** | **Strategies** | |
| 1 | Informed Search | Admissible Heuristics |
| 2 | Inadmissible Heuristics |
| 3 | Learning | Approximate Function |
| 4 | Generating Data |
| 5 | Compare the results | |

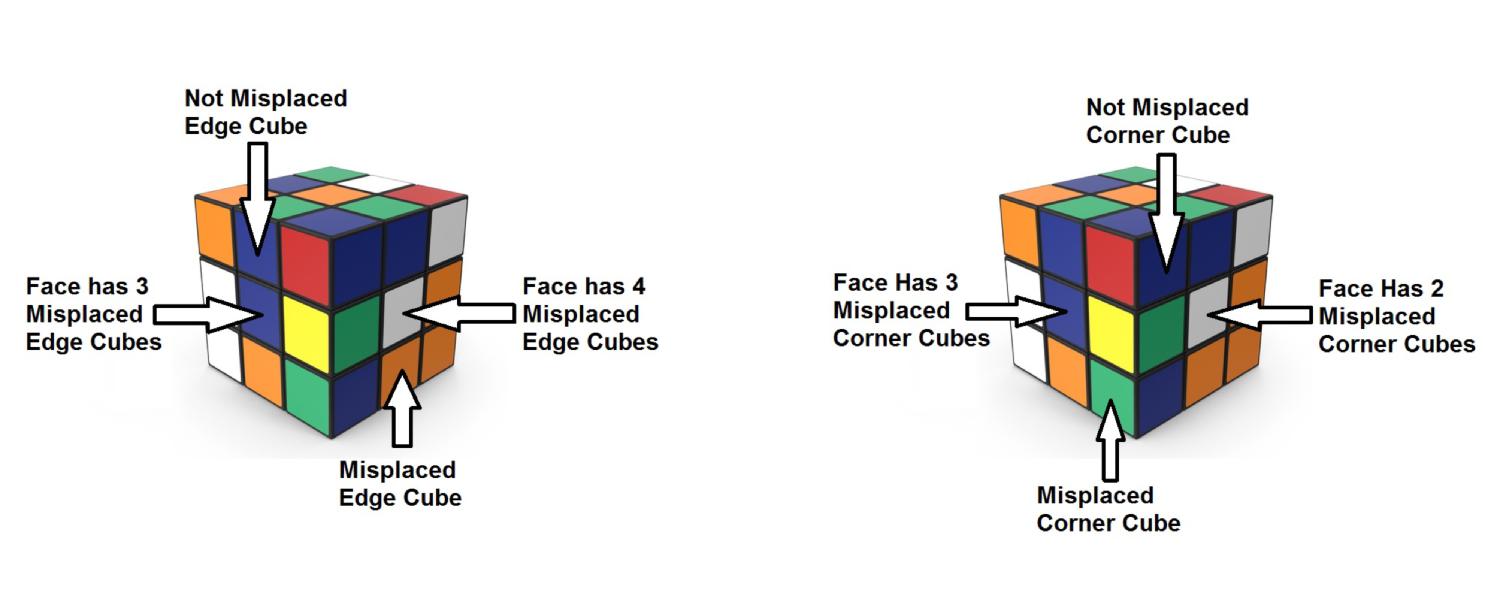
* + - 1. Admissible Heuristics

1. Max Misplaced Edge Cubes in Face

In a 3 by 3 cube (and in any other odd dimension cube) the colour of a face can be categorized as the colour of the centre cube. For an edge cube, we say that it is misdirected if the colour is different from the centre cube (refer Figure 1.4: Cube notations for heuristics (a)). For each face we add more than the number of wrong edge cubes. Then we take the total over the faces This heuristic is acceptable because in a solved cube the edge cubes has to be the same colour as the centre cube and we need at least one action to " fix " each one of them. This heuristic is larger than 0 and is limited to 4. In mathematical notation:



**Figure 1.3: Max Misplaced Edge Cubes in Face Formula.**

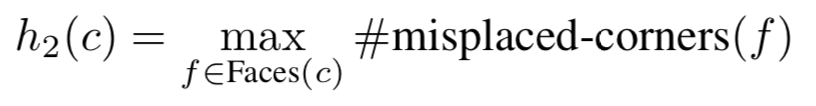


(a) Misplaced Edges (b) Misplaced Corners

**Figure 1.4: Cube notations for heuristics**

1. Max Misplaced Corner Cubes in Face

For a corner cube, we conclude it is misplaced if it is a different colour from the two adjacent edge cubes (see Figure II.1b). For each face, we add more than the number of corner cubes. We then take over all faces the maximum of this quantity. This heuristic is acceptable because the corner cubes must be the same colour as the adjacent edge cubes, we need at least one action to put them in place. This heuristic is larger than 0 and is limited to 4.

****

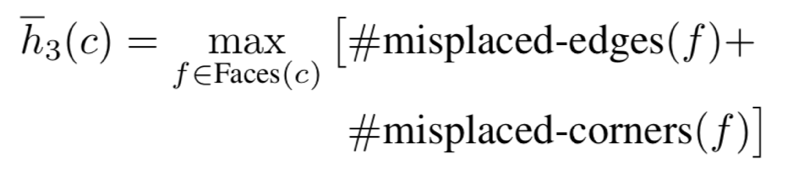
**Figure 1.5: Max Misplaced Corner Cubes in Face Formula.**

* + - 1. Inadmissible Heuristics

The allowable heuristics are limited by a value of 4, which is wholly insufficient for scrambles of 5 moves or beyond, so we also consider the following inadmissible heuristics which can return larger (significant) values.

1. Max Misplaced Corner and Edge Cubes in Face

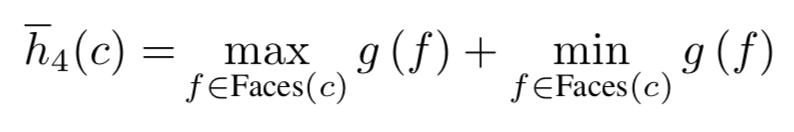
For a face we have over the number of wrong corner cubes and edge cubes. Then we take the maximum across the faces. This heuristic is not allowed. This Heuristic is greater than 0 for an unresolved cube and limited to 8.



**Figure 1.6: Max Misplaced Corner and Edge Cubes in Face Formula.**

1. Max Plus Min Misplaced Corner and Edge Cubes in Face

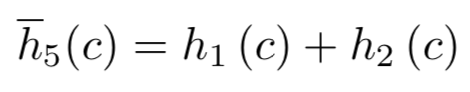
For a face we have over the number of wrong corner cubes and edge cubes. Then we take the maximum over our faces and the minimum over our faces. This heuristic is not permitted. This Heuristic is larger than 0 and is limited to 16 .



**Figure 1.7: Max Plus Min Misplaced Corner and Edge Cubes in Face Formula.**

1. Max Misplaced Corner Cubes Plus Max Misplaced Edge Cubes in Face

For a face we have over the number of wrong corner cubes and edge cubes. Then we take the maximum over our faces and the minimum over our faces. This heuristic is not permitted. This Heuristic is greater than 0 for an unresolved cube and limited to 8.



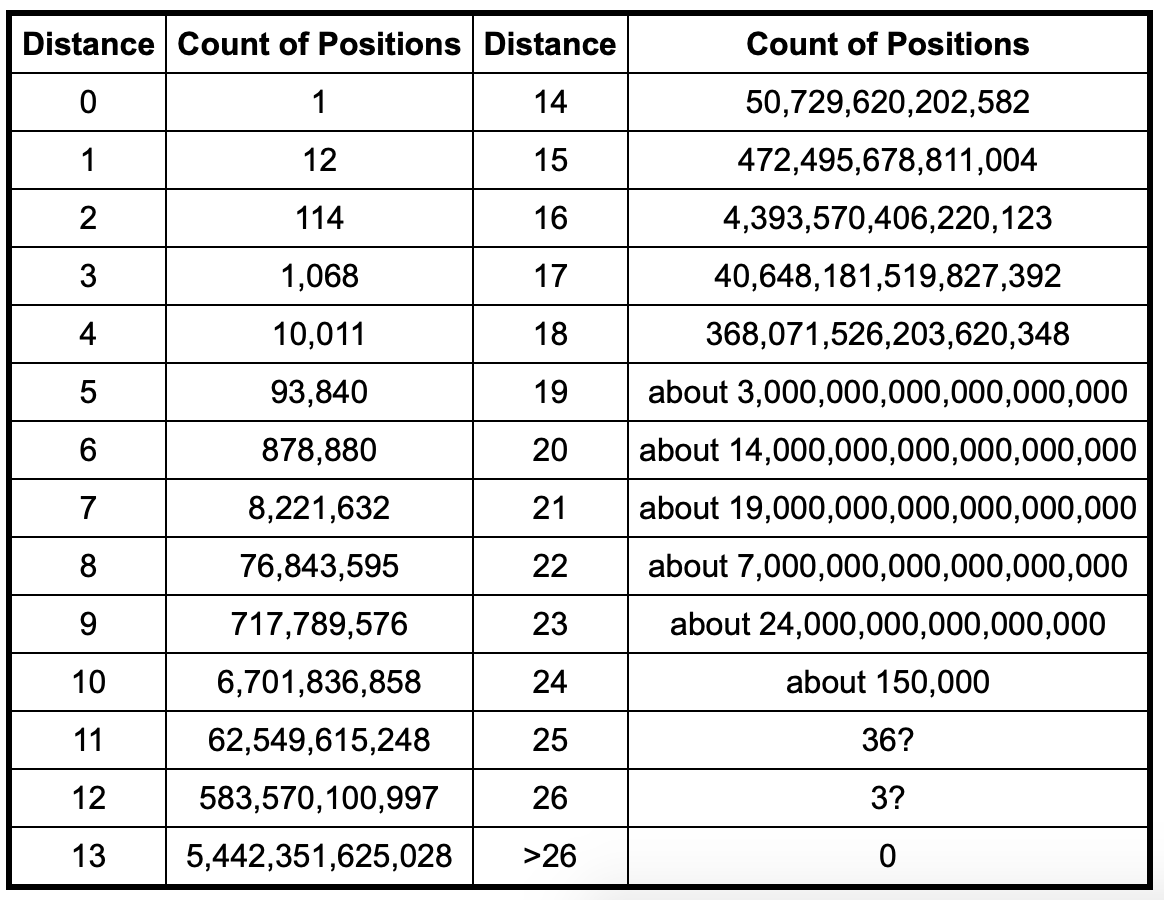
**Figure 1.8: Max Misplaced Corner Cubes Plus Max Misplaced Edge Cubes in Face Formula.**

###### Train the model

1. Approximate Function

Let C be the space of all 3 by3 cubes. We aim to train a model that works closely with h: c! N that maps every cube setup to its shortest distance from the target state. To achieve this, we decided to train a regression model of the Deep Neural Network in a supervised learning setup.

Deep Neural Networks (DNN): We know that any other continuous function over compact subsets of Rn can be combined with arbitrary precision using a neural feed network (B. Csáji, 2001). Theoretically, neural networks have more than one layer hidden (i.e. Deep networks) tend to do better than shallow networks. We termed several net architectures and parameters with more than three hidden layers fully connected.



**Figure 1.8 Histogram of cube configurations.**

1. **Generating Data**

For the training of the estimator, it is essential to generate examples with a distance from the target. We had to generate mixed cube data examples with distance d. To do this, we have taken a random step back from a target state. This gives us a cube setting of distance d, as we cannot be sure that there are no shorter walks with the same result. The number of configurations with a given distance increases exponentially with distance (up to d 20, refer figure 1.8 Histogram of cube configurations), so intuitively the probability of obtaining a distance configuration d (and not one with a smaller distance) should be high by performing d random movements. You may need to reason about the mixing time of the random walk on the configuration graph to make this claim rigorous. When we looked at it, we decided that it was beyond the scope of this project. In order to train the estimator, we need to generate a lot of random cube configurations (80; 000) per training session, so that even if the above is present, shorter than intended configurations can still arise.

###### Compare the results

A reasonable measurement of the performance of different heuristics is the number of nodes that the A\* algorithm expanded during its run. For the human-made heuristics, this is the only measurement we got. For the learned estimators, we considered the Mean Square Error (MSE) of its predictions as another indicator for the model’s performance.

# REFERENCE

Boris Gorshenev, Progress Report Rubik’s Cube, CMPS 140 - Spring 2018, 2018

D. Silver, A. Huang, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016.

E. D. Demaine, S. Eisenstat, and M. Rudoy. Solving the Rubik’s Cube Optimally is NP-complete. ArXiv e-prints, June 2017.

Erez Karpas and Carmel Domshlak. Optimal search with inadmissible heuristics. In Proceedings of International Conference on Automated Planning and Scheduling, 2012.

Francisco Javier Ordóñez and Daniel Roggen. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition, 2016.

Irwan Bello, HieuPham, QuocV.Le, MohammadNorouzi, and Samy Bengio. Neural Combinatorial Optimization , November, 2016.

Jordan Thayer, Austin Dionne, and Wheeler Ruml. Learning inadmissible heuristics during search. In Proceedings of International Conference on Automated Planning and Scheduling, 2011.

Leedy, P. D., Practical research: planning and design. New Jersey: Prentice-Hall, 1993.

Matej Moravcˇík et al. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science, 2017.

Mehdi Samadi, Ariel Felner, and Jonathan Schaeffer. Learning from multiple heuristics, 2016.

Nathan R.Sturtevant, Ariel Felner, and MalteHelmert. Exploiting the rubik’s cube 12-edge PDB by combining partial pattern databases and bloom filters. In Stefan Edelkamp and Roman Barták, editors, Proceedings of the Seventh Annual Symposium on Combinatorial Search, SOCS 2014. AAAI Press, 2014.

Stephen McAleer, Forest Agostinelli, Alexander Shmakov, Pierre BaldiStephen McAleer, Forest Agostinelli, Alexander Shmakov, Pierre Baldi. Solving the Rubik's Cube Without Human Knowledge, 2018.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 Edition, 2010.

Thomas Keller, Florian Pommerening, Jendrik Seipp, Florian Geißer, and Robert Mattmüller. State-dependent cost partitioning’s for cartesian abstractions in classical planning. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016), 2016.

Youssef Hamadi, Eric Monfroy, Springer-Verlag, and Frédéric Saubion. Autonomous search, 2012.

B. Csáji, “Approximation with artificial neural networks,” *MSc. thesis*, 2001.