**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](#_Toc8382056)

[CHAPTER 1 5](#_Toc8382057)

[INTRODUCTION 5](#_Toc8382058)

[1.1 Research Background 5](#_Toc8382059)

[1.2 Motivation 6](#_Toc8382060)

[1.3 Research Problem 7](#_Toc8382061)

[1.4 Objective 7](#_Toc8382062)

[1.5 Research Question 7](#_Toc8382063)

[1.6 Research Delivery 8](#_Toc8382064)

[1.7 Research Signification 8](#_Toc8382065)

[CHAPTER 2 9](#_Toc8382066)

[LITERATURE REVIEW 9](#_Toc8382067)

[2.1 Introduction 9](#_Toc8382068)

[2.1.1. Pattern Database Learning 10](#_Toc8382069)

[2.1.2. Baseline Learning 11](#_Toc8382070)

[2.1.3. Boosting Learning 12](#_Toc8382071)

[2.1.4. Autodidactic Iteration 12](#_Toc8382072)

[2.1.5. Heuristic Learning 13](#_Toc8382073)

[2.1.6. Comparation of Learning 14](#_Toc8382074)

[CHAPTER 3 16](#_Toc8382075)

[RESEARCH METHODOLOGY 16](#_Toc8382076)

[3.1 Quantitative Research 16](#_Toc8382077)

[3.2 Informed Search Algorithm 17](#_Toc8382078)

[3.2.1 Admissible Heuristics 17](#_Toc8382079)

[3.2.1.1 Max Misplaced Edge Cubes in Face 17](#_Toc8382080)

[3.2.1.2 Max Misplaced Corner Cubes in Face 18](#_Toc8382081)

[3.2.2 Inadmissible Heuristics 19](#_Toc8382082)

[3.2.2.1 Max Misplaced Corner and Edge Cubes in Face 19](#_Toc8382083)

[3.2.2.2 Max Plus Min Misplaced Corner and Edge Cubes in Face 19](#_Toc8382084)

[3.2.2.3 Max Misplaced Corner Plus Max Misplaced Edge Cube in Face 20](#_Toc8382085)

[3.3 Training Models 20](#_Toc8382086)

[3.3.1 Approximate Function 20](#_Toc8382087)

[3.3.2 Generating Sample Data 21](#_Toc8382088)

[3.3.3 Allocation for Choosing the 22](#_Toc8382089)

[3.3.4 Compare Results 23](#_Toc8382090)

[CHAPTER 4 24](#_Toc8382091)

[RESULT AND ANALYSIS 24](#_Toc8382092)

[4.1 Python Scripts 24](#_Toc8382093)

[4.2 Results 25](#_Toc8382094)

[4.2.1 Admissible Over Inadmissible 25](#_Toc8382095)

[4.2.2 Comparing Learned Heuristics 25](#_Toc8382096)

[4.2.3 Learned Over Human-Made Heuristics 28](#_Toc8382097)

[CONCLUSION AND FUTURE RESEARCH 29](#_Toc8382098)

[5.1 Conclusion 29](#_Toc8382099)

[5.2 Future Research 29](#_Toc8382100)

[5.2.1 Improve Learning Model 29](#_Toc8382101)

[5.2.2 Benchmark on other Heuristic Models 29](#_Toc8382102)

[REFERENCE 30](#_Toc8382103)

# 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 the 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 who were 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 vital 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 straightforward and are not finetuned for the specific problem.

With the recent rapid development of deep learning models, many new possibilities are now available in this area. Learning algorithms now exist for practical 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 specific algorithms. Furthermore, the fact that there are distinct 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 several 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 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 a tremendous amount of mathematical logic to develop new solutions, it is challenging 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 are 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 an 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.

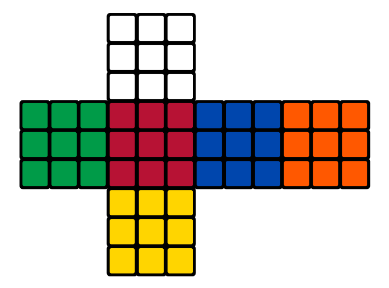
### Research Signification

* 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 REVIEW

### Introduction



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

Rubik’s cube, the incredible riddle, may fill in for instance of an arranging issue as its undertaking is to discover an arrangement of activities driving structure given beginning state to a predefined objective state. From a search perspective, the number of different states of the standard 3 by 3 cube is , which are over 43 quintillion legal positions of the Rubik’s Cube, to be specific .

At approximate 26 quarter-moves, every state can be solved optimally. There is a single objective state, and the fanning factor is 12 when using symmetry breaking representation — considering a model utilizing quarter-move meaning that it is permitted to turn layers just by 90◦. The half-move portrayal enables sides to be pivoted by 180◦ in a single move.

To be able to describe some different properties of the cube, firstly need to define a couple of additional thoughts. The cubies solid shape comprises of 27 little blocks which are known as cubies. Some faces of these cubies are coloured, or more precisely they carry coloured stickers. In the objective express, all appearances of the substantial 3 by 3 shape contain just stickers of a similar shading.

In the event dismantled the cube-square, by getting 8 corner- cubies, each of which carries 3 coloured stickers, 12 edge- cubies, each with 2 stickers on them and 6 central-cubies each having 1 colour.

A cube shape of any size can without much of a stretch be understood not well utilizing a straightforward calculation that runs in time , where n is the size of the cube. Finding optimal solutions seems to be substantially harder, even though 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, if I consider the 8 corner cubies of a standard 3 by 3 cubes. The entire state space of this smaller issue might be listed, solved optimally for each state and stored in a database. For each condition of the standard 3D shape, it will be capable for me broaden it on the precedent by disregarding everything except the corners, look-into this state in the database and utilize its assessment as a lower bound on the length of the arrangement. The set of features that were considered in the smaller problem is called pattern, and features 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 expresses, the heuristic esteem is 0, and the calculation encounters significant troubles finding the real state since it has no further direction. For proficient looking, it is essential to join a few PDBs with various examples or to consolidate PDB with different kinds of heuristics. There are numerous methods for joining the heuristics, from basic ones, such as taking most significant, to progressively complex ones like added substance PDBs or cost dividing (Florian Pommerening, 2016).

#### Baseline Learning

Baseline method, neural system design for the frail student. All structures tried were prepared on the errand of foreseeing the right move succession when given a grouping of square cube states. To encode each cube shape express, the scientist utilizes a one-hot encoding of sticker hues. To encode each cube state, the researcher uses a one-hot encoding of sticker colours. There are 54 colours, and each can be one of 6 colours. Indicator variable is 1 if the sticker is the color, and 0 otherwise. This gives an aggregate of 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 more comfortable to implement, and initial results were promising enough to continue further work.

Nevertheless, a cube representation may be worth pursuing. Each pattern engineering utilized finishes with 12 yields, compared to the 12 conceivable face turns - every one of the 6 appearances can be pivoted clockwise or against 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 calculations are a general class of group strategies that convert a few powerless students into a solitary solid one. They have hypothetical certifications on assembly rates for mistake however every so often bomb by and by. (Bengio, Yoshua, 2009)

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

#### Autodidactic Iteration

Autodidactic Iteration is an iterative monitored learning procedure that trains a deep network of neural parameters for the input state and the output of a value and policy pair (). The policy output p is a vector with the probabilities to move from the state for each of the 12 moves. The policy will be used to decrease breadth and the value to reduce depth in the MCTS once the network is trained. Training samples for are generated from the solved cube for each Autodidactic Iteration. This ensures that specific training inputs are sufficiently close to have a positive reward for a shallow search. Targets will then be created by conducting the first-width breadth search (BFS) of each sample of training. To estimate the value of each child, the current value network is used. The maximum value and reward for each sample of their children is the value target, and the policy objective is the action leading to this maximum value. (Stephen McAleer, 2018)

#### Heuristic Learning

The task of heuristic learning is to consequently create a heuristic function for given problem based on some training data. The learning has usually completed a priory, where the training data are provided before the search. The 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).

A few endeavours have been made to use Neural Networks for the heuristic learning task (Austin Dionne, 2011). In the standard setting, a course of action of highlights is figured for each state in the preparation set just as the ideal separation to-go to the closest physical state and the system is then used to take in mapping from highlights to remove the gauge. After the learning methodology is done, the framework is used as a heuristic separation estimator together with an informed forward search algorithm like or \*.

Heuristics that are found out along these lines give 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 does not typically guarantee to find optimal solutions, this approach is just reasonable in situations where close-to-optimal solutions are sufficient. It is, in any case, conceivable to ensure optimality even with a prohibited heuristic by adjusting the inquiry strategy (Erez Karpas and Carmel Domshlak, 2012).

From the intricacy point of view, the assignment of heuristic learning, by and large, is hard. It is realized that the undertaking of discovering perfect responses to some arranging issues like summed up 15-puzzle, Sokoban, and a lot more is NP-hard or considerably harder. It is also known that with an accurate-enough heuristic, the search time may be polynomial. To be specific, if 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 evident 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 significant problems optimally in polynomial time, but it may still outperform classic human-designed heuristics.

#### Comparation of Learning

**Table 1.2: Comparation of Learnings**

|  |  |
| --- | --- |
| **Learning** | **Differences** |
| **Pattern Database Learning** | A pattern is the partial specification of a permutation (or state); that is, the tiles occupying specific locations are unspecified. |
| **Baseline Learning** | A baseline prediction algorithm gives many expectations that can assess as would any expectations for the issue. |
| **Boosting Learning** | 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 to be decently exact on the preparation set. |
| **Autodidactic Iteration** | Autodidactic Iteration is a new learning algorithm for strengthening that can teach itself how to solve the problem without human assistance. After training, the network then uses a standard search tree to search for each configuration for suggested movements. |
| **Heuristic Learning** | A heuristic is a dependable general guideline, generally created through understanding, that is joined into necessary leadership process keeping in mind the end goal to tackle an issue. Numerous charts look calculations, for example, , utilize the heuristic-based pursuit |

# CHAPTER 3

## RESEARCH METHODOLOGY

### Quantitative Research

Quantitative research techniques are examine strategies managing numbers and anything quantifiable in an orderly method for examination of wonders and their connections. It is applied when a question is answered based on relationships within measurable variables to explain, predict and control a phenomenon (Leedy 1993).

A complete quantitative examination usually finishes with affirmation or disconfirmation of the speculation tried. Specialists utilizing the quantitative strategy distinguish one or a couple of factors that they plan to use in their examination work and continue with information accumulation identified with those factors.

Using a scientific approach, quantitative methods often deal with results computation and system analysis in the field of Information Technology (IT). The target of the quantitative technique is to create and utilize models dependent on numerical methodology, speculations and hypotheses about the idea 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 an iterative process where evidence is evaluated, and hypotheses and theories are refined with some technical advances, leveraging on a statistical approach.

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

For an example, when a researcher is interested in investigating the “*effectiveness of the 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.

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

**Table 1.3: Strategies used in observation studies.**

#### Admissible Heuristics

##### Max Misplaced Edge Cubes in Face

In a 3 by 3 cube the colour of a face can be categorized as the colour of the centre cube. For an edge cube, it is said to be misdirected if the colour is different from the centre cube (*refer Figure 1.2: Cube notations for Misplaced Edges Cube*). For each face, more than the number of wrong edge cubes is added. Then the total is taken over the faces.

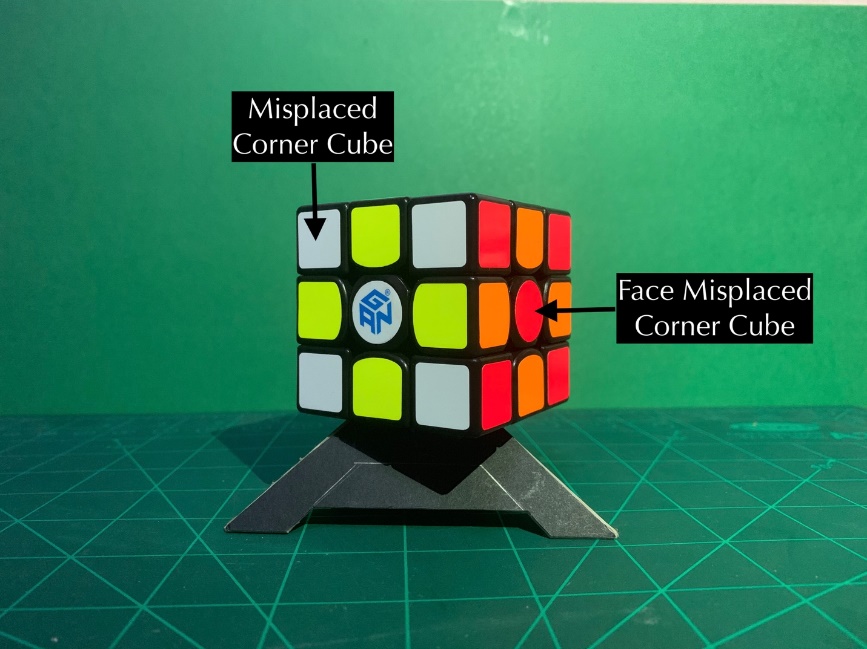
This heuristic is acceptable because in a solved cube the edge cubes has to be the same colour as the centre cube and at least one action is needed to "fix" each one of them. This heuristic is larger than 0 and is limited to 4. In mathematical notation:

****

**Figure 1.2: Cube notations for Misplaced Edges Cube.**

##### Max Misplaced Corner Cubes in Face

For a corner cube, it is concluded it is misplaced if it is a different colour from the two adjacent edge cubes (*refer Figure 1.3: Cube notations for Misplaced Corner Cube*). For each face, more is added than the number of corner cubes. It then takes 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, at least one action is needed to put them in place. This heuristic is larger than 0 and is limited to 4. In mathematical notation:

****

**Figure 1.3: Cube notations for Misplaced Corner Cube.**

#### Inadmissible Heuristics

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

##### Max Misplaced Corner and Edge Cubes in Face

For a face it must have over the number of wrong corner cubes and edge cubes. Then 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. In mathematical notation:

##### Max Plus Min Misplaced Corner and Edge Cubes in Face

For a face it has over the number of wrong corner cubes and edge cubes. Then the maximum is taken over the faces and the minimum over the faces. This heuristic is not permitted. This Heuristic is larger than 0 and is limited to 16. In mathematical notation:

##### Max Misplaced Corner Plus Max Misplaced Edge Cube in Face

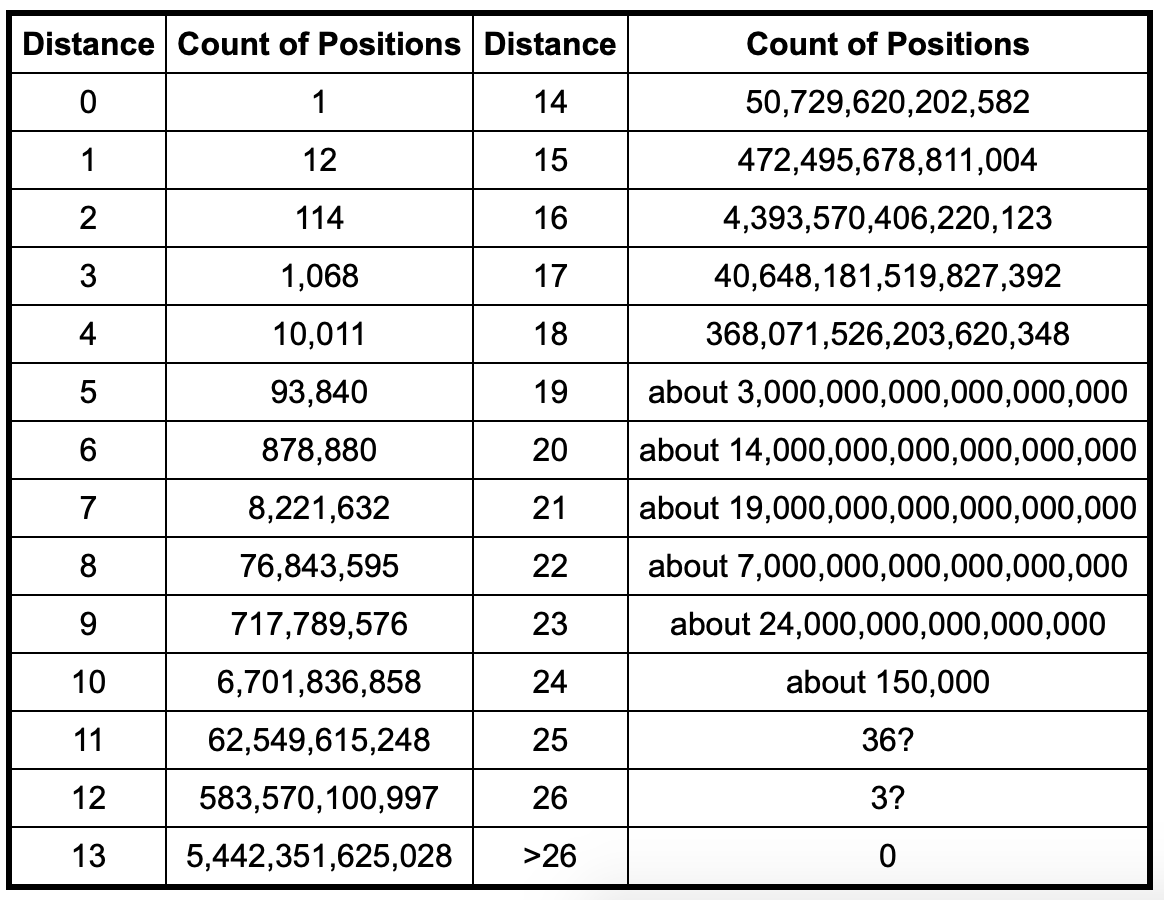
For a face it has over the number of wrong corner cubes and edge cubes. Then the maximum is taken over the faces and the minimum over the faces. This heuristic is not permitted. This Heuristic is greater than 0 for an unresolved cube and limited to 8. In mathematical notation:

### Training Models

#### Approximate Function

Let be the space of all 3 by 3 cubes. The aim is to train a model that works closely with that maps every cube setup to its shortest distance from the target state. To achieve this, it is decided to train a regression model of the Deep Neural Network in a supervised learning setup.

Deep Neural Networks (DNN): It is known 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 tend to do better than shallow networks. It is termed several net architectures and parameters with more than three hidden layers fully connected.



**Figure 1.4 Histogram of cube configurations.**

#### Generating Sample Data

For the estimator training, it is essential to generate examples with a given distance from the objective — data examples of mixed cubes with distanced needed to be generated. In order to do this, went back randomly from an objective state. That provides a cube setup that is long, as it cannot be sure that there are no shorter walks that will produce the same result.

With the distance (up to , *refer Table 1.4:* *Average solution length, taken over 100 examples for each .),* it is necessary to increase exponentially the number of configurations with a given distance, so that intuitively the probability of a configuration of distance d (not one with a smaller distance) should be high. In order to train the estimator, it must be generated numerous configurations of random cubes (almost 80,000) per training session, so that although the above is present, configurations may remain shorter than intended.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 7 | 8 | 9 |
|  | 7.02 | 8 | 9 |
|  | 7.02 | 8 | 9.14 |
|  | 7.02 | 8 | 9 |

**Table 1.4: Average solution length, taken over 100 examples for each .**

Used three methods to remove these cases:

1. (Trivial) Keep a set of visited states and make sure that I don't go to an already visited state during the walk.
2. Three or more identical movements disallow. The sequence for example is limited to .
3. If they commute, two actions are independent. Independent actions, in this case, are actions that operate on the face of the other cube (the front and the back). Later decide to order those actions (the front is smaller than the back) for any two independent actions. If two next actions are independent, the smaller one must come first in any given sequence of actions. For this purpose, generating a more extended sequence of measures in the random walk and then use a variant of Bubble Sort to exchange inappropriate actions. Finally, used the previous steps to reduce redundant measures after sorting the sequence.

#### Allocation for Choosing the

As noted, each example of training is produced by selecting the length of the random path, indicated by d first and then creating an altered sequence of measures of length . As the number of configurations grows exponentially, it does not make much sense to uniformly choose from a set { }. In fact, the learned model was not performable on larger distances after trying to train this way.

On the other hand, the choice of does not work well either according to the "actual" configuration allocation. Almost every time the value is selected, and the model does not perform well in short-distance settings. Such a model could not be used in as a solved cube is not recognized.

The solution was for the two distributions to be combined convexly. The point to from the set total for a single uniform distribution, both over { }, which is a distribution valid. Finally, chosen to work best.

#### Compare Results

The number of nodes expanded to include the algorithm during operation is reasonable in the performance of various heuristics. This is the only measure have for heuristics made by humans. In the learned estimators, the Mean Square Error (MSE) of its estimates was considered another indicator of model performance.

# CHAPTER 4

## RESULT AND ANALYSIS

Performed various tests to compare different heuristics. First, compared regular heuristics in comparison and compare the heuristics that have learned. Finally, compared the best heuristic from the first group to the next group.

### Python Scripts

The script is split into 2 main parts:

1. **The cube module** – Consists of a simple class representing a cube, the Rubik’s Cube problem written as an Search Problem and some other useful functions. This code provided by the author of “*Solving the Rubik's Cube Without Human Knowledge, 2018*”, for benchmarking (Forest Agostinelli, 2018).

PyPI package link: https://pypi.org/project/rubikai/

1. **Jupyter Notebooks 3** – Virtually hosted on the Google Collaboratory, which contain two parts:
2. Model training:

https://drive.google.com/open?id=1gZYofPDtigzCc85CpHg83Of20pYNJR6q

https://drive.google.com/open?id=1cV8UvUwzCxwJ0DK8qVS0xkJ61IPFCLa4

1. Main notebook:

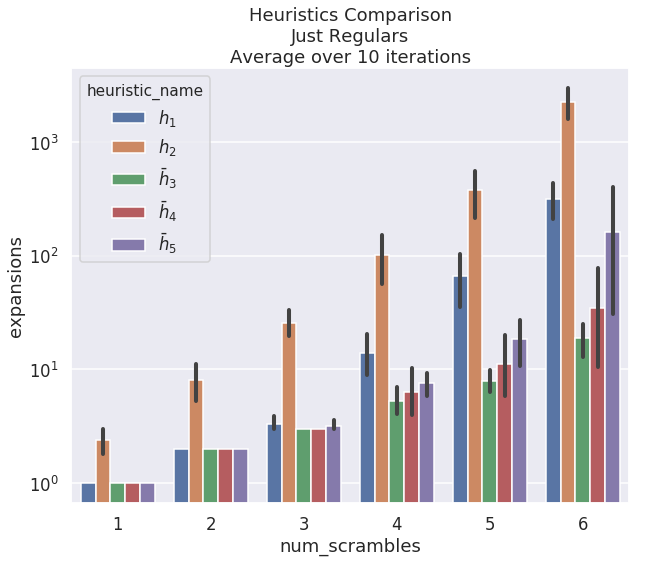
<https://drive.google.com/open?id=1fklG8Q4MY1POz_a2KXOWuLR8qp-UczFa>

### Results

#### Admissible Over Inadmissible

As anticipated, the heuristics allowed for (*refer Table 1.5: Result of Heuristics Comparison Just Regulars of Average over 10 iterations.*) were too inefficient, so only used the inadmissible values for larger values. These actions were successful in solving up to = 25 moves. The heuristic has been especially successful with averages of 1, 500 nodes for 25 scramble movements (*refer Figure 1.5: Result Heuristics Comparison Just Regulars of Average over 10 iterations*).

Furthermore, unacceptable heuristics did not significantly affect the optimal solution. To test this, checked whether the returned length of the solution exceeds the number of scramble moves for this example. The results are shown in *Table 1.4: Average solution length, taken over 100 examples for each d.*, where for scramble = 1 to 6, the solutions returned have always been optimum.



**Figure 1.5: Result of Heuristics Comparison Just Regulars of Average over 10 iterations.**

#### Comparing Learned Heuristics

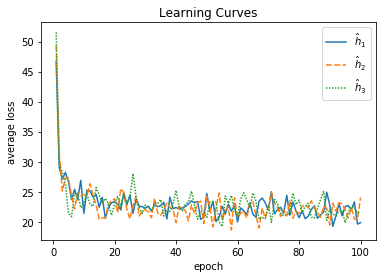
Three models are trained for a DNN regression with the different layer and neuron numbers:

: has three layers, of 70, 60 and layer of 50 neurons, respectively.

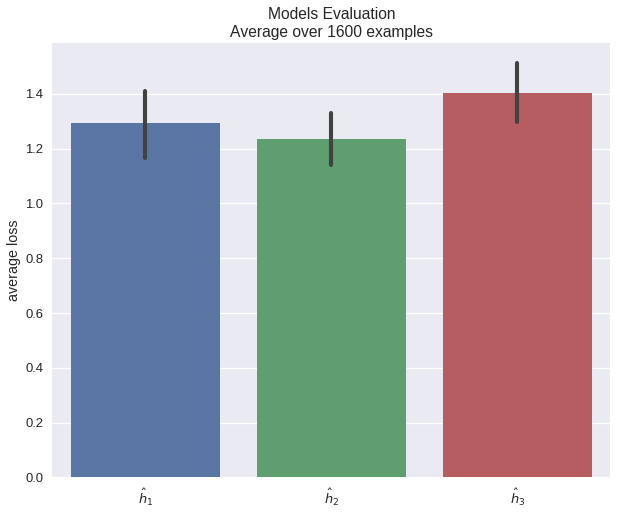
: has four layers and 50 neuron levels each layer.

: 5 layers, with neurons: 50, 40 and 30. 20.

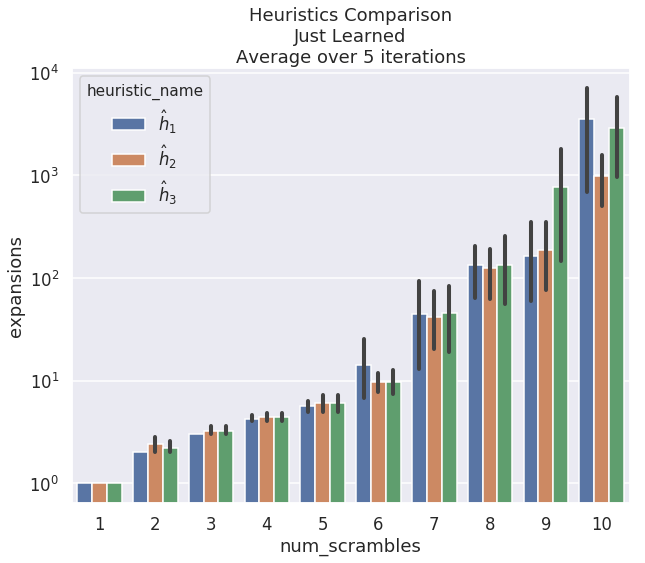
All models have been trained on 100,000 labelled cube examples, with about the same loss and training curves (*refer Figure 1.6: Learning Curves of Training Data for Plateau from 100 to 200 Epoch*.). It seems as if the learning curve reaches a plateau and stopped training because the assessment loss is similar (*refer Figure 1.7: Model Evaluation of Average over 1600 examples.),* the evaluation loss was compared to the heuristics (*refer Figure 1.8: Result of Heuristics Comparison Just Learned of Average over 5 iterations*.).



**Figure 1.6: Learning Curves of Training Data for Plateau from 100 to 200 Epoch.**

****

**Figure 1.7: Model Evaluation of Average over 1600 examples.**

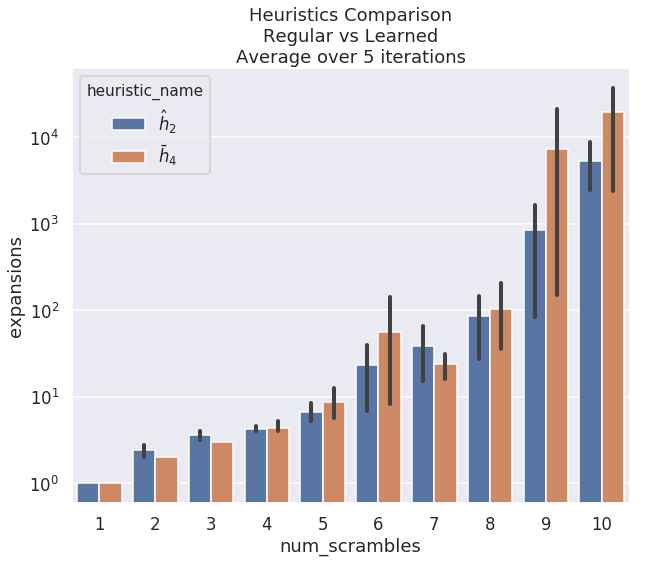
****

**Figure 1.8: Result of Heuristics Comparison Just Learned of Average over 5 iterations.**

#### Learned Over Human-Made Heuristics

The learned heuristic, , is marginally better than the best person made, , as shown in the results (*refer to Figure 1.9: Result of Heuristics Comparison Regular Versus Learned of Average over 5 iterations.*).

During the assessment of the learned evaluator, it was relatively accurate, and expected it to perform better. However, the results satisfied with the results given the limited training time and data.

****

**Figure 1.9: Result of Heuristics Comparison Regular Versus Learned of Average over 5 iterations.**

# CONCLUSION AND FUTURE RESEARCH

### Conclusion

The learned heuristics were generally perfect compared to those produced by humans. In particular, as the learning process has become limited, there have been relatively few parameters tweaked, and not more than two hours of training time per model, which is significantly less than other Rubik’s Cube solvers. (Stephen McAleer, 2018)

Note that the training process with large amounts of resources can be performed once in this proposed context, and the former model can then be applied by a low-end machine to solve problems effectively.

### Future Research

#### Improve Learning Model

This project was very timely and computational and was the first real experience with the use of neural web frameworks. Finally, believe that further architecture and tweaking of hyperparameters will improve the performance of models.

#### Benchmark on other Heuristic Models

The heuristic that has compared to those in this research. There are well-researched heuristics that can compete more effectively with the improved learning heuristics Korf’s Algorithm (H.Kaur, 2015), which can be used to improve the learned heuristic.

# REFERENCE

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

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.

H. Kaur, “Algorithms for solving the Rubik’s cube,” Tech. Rep., 2015.

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.