NUERAL RUBIK'S – SOLVING RUBIK'S CUBE USING NEURAL NETWORK (HEURISTIC LEARNING)

GUNASEGARRAN MAGADEVAN

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF DATA SCIENCE

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Gunasegarran Magadevan (I.C/Passport No: 920630146453)

Matric No: WQD170002

Name of Degree: Master of Data Science

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

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

Field of Study: Neural Network (Heuristic Learning)

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidature's Signature Date: 23 May 2019

Subscribed and solemnly declared before,

Witness's Signature: Date: 23 May 2019

Name: Dr. Aznul Qalid Md Sabri
Designation: Supervisor

UNIVERSITY OF MALAYA

PERAKUAN KEASLIAN PENULISAN

Nama: Gunasegarran Magadevan (No. K.P/Pasport: 920630146454)

No. Matrik: WQD17002

Nama Ijazah: Master of Data Science

Tajuk Kertas Projek/Laporan Penyelidikan/Disertasi/Tesis ("Hasil Kerja ini"):

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

Bidang Penyelidikan: Neural Network (Heuristic Learning)

Saya dengan sesungguhnya dan sebenarnya mengaku bahawa:

(1) Saya adalah satu-satunya pengarang/penulis Hasil Kerja ini;
(2) Hasil Kerja ini adalah asli;
(3) Apa-apa penggunaan mana-mana hasil kerja yang mengandungi hakcipta telah dilakukan secara urusan yang wajar dan bagi maksud yang dibenarkan dan apa-apa petikan, ekstrak, rujukan atau pengeluaran semula daripada atau kepada mana-mana hasil kerja yang mengandungi hakcipta telah dinyatakan dengan sejelasnya dan secukupnya dan satu pengiktirafan tajuk hasil kerja tersebut dan pengarang/penulisnya telah dilakukan di dalam Hasil Kerja ini;
(4) Saya tidak mempunyai apa-apa pengetahuan sebenar atau patut semunasabahnya tahu bahawa penghasilan Hasil Kerja ini melanggar suatu hakcipta hasil kerja yang lain;
(5) Saya dengan ini menyerahkan kesemua dan tiap-tiap hak yang terkandung di dalam hakcipta Hasil Kerja ini kepada Universiti Malaya ("UM") yang seterusnya mula dari sekarang adalah tuan punya kepada hakcipta di dalam Hasil Kerja ini dan apa-apa pengeluaran semula atau penggunaan dalam apa jua bentuk atau dengan apa juga cara sekalipun adalah dilarang tanpa terlebih dahulu mendapat kebenaran bertulis dari UM;
(6) Saya sedar sepenuhnya sekiranya dalam masa penghasilan Hasil Kerja ini saya telah melanggar suatu hakcipta hasil kerja yang lain sama ada dengan niat atau sebaliknya, saya boleh dikenakan tindakan undang-undang atau apa-apa tindakan lain sebagaimana yang diputuskan oleh UM.

Tandatangan Calon Tarikh: 23 May 2019

Diperbuat dan sesungguhnya diakui di hadapan,

Tandatangan Saksi Tarikh: 23 May 2019

Nama: Dr. Aznul Qalid Md Sabri

Jawatan: Supervisor

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

ABSTRACT

Neural networks have already proven to be able to cope with noisy and unstructured data like hand-written texts, images, sounds and classification of real-world objects based on the incomplete description. 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. Two types of heuristics algorithm are utilized, firstly, an admissible heuristic or informed search is utilized to assess the expense of achieving the objective state in the heuristic algorithm. In this research, the objective state would be for the Rubik's Cube to be in a solved state. All together for a heuristic to be permissible to heuristic, the assessed cost should dependably be lower than or equivalent to the actual cost of achieving the objective state. The heuristic algorithm utilizes the admissible heuristic to discover an expected ideal way to the objective state from the current state. While an inadmissible heuristic, means that it might sometimes find the non-optimal path. By biasing the estimate, it can choose between path cost and search cost. 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. Implementing neural network approaches to solve the Rubik's Cube have struggled to succeed without human help and have had to rely on hand-engineered features and group theory to systematically find solutions. The finding of this research paper is to automatically obtain valuable heuristic functions so that the entire problem-solving process is free of human knowledge. This also helps to use heuristic algorithm rules that perform random walks back from a goal state and try to learn how far they have reached the goal. Several tests ran to compare the different heuristics. Firstly, comparison is done between the regular heuristics (human-made heuristic, the trained model using heuristic rules), with each other. Then, comparison is done between the learned heuristics (the learned model from regular heuristic), with each other. Finally, compared the best heuristic from the first group to the respective one from the second group. The results are analyzed as the performance of different heuristics is reasonable. This is due to the fact that number of nodes are expanded to include A^* algorithm.

Keywords: Neural Network, Rubik's cube, Heuristic learning, Admissible Heuristics, Inadmissible.

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

ABSTRAK

Neural Networks telah terbukti dapat menampung data bising dan tidak berstruktur seperti teks, imej, bunyi dan klasifikasi objek dunia nyata berdasarkan perihalan yang tidak lengkap. Sesetengah perhatian juga telah didedikasikan untuk pembelajaran heuristik, di mana tugasnya secara automatik mendorong fungsi heuristik dari contoh latihan menggunakan model pembelajaran mesin. Dua jenis algoritma heuristik digunakan, pertama, heuristik yang boleh diterima atau pencarian maklumat digunakan untuk menilai perbelanjaan mencapai objektif objektif dalam algoritma heuristic. Dalam kajian ini, keadaan objektif adalah untuk Rubik's Cube berada dalam keadaan yang diselesaikan. Semua bersama-sama untuk heuristik yang dibenarkan untuk heuristik, kos yang dinilai harus bergantung lebih rendah daripada atau setara dengan kos sebenar untuk mencapai keadaan objektif. Algoritma heuristik menggunakan heuristik sdmissible untuk mengetahui cara ideal yang diharapkan untuk keadaan objektif dari keadaan semasa. Walaupun heuristik inadmissible, ia mungkin kadang-kadang mencari laluan yang tidak optimum. Dengan membiasakan anggaran, ia boleh memilih antara kos laluan dan kos carian. Rubik's Cube menggunakan teori kumpulan matematik, yang telah membantu menyimpulkan algoritma tertentu. Selain itu, hakikat bahawa terdapat subkumpulan yang berbeza dalam kumpulan Rubik Cube membolehkan teka-teki itu dipelajari dan dikuasai dengan bergerak melalui "tahap kesukaran" yang berbeza. Melaksanakan pendekatan rangkaian saraf untuk menyelesaikan Rubik's Cube telah berjuang untuk berjaya tanpa bantuan manusia dan harus bergantung pada ciri-ciri tangan dan teori kumpulan untuk mencari penyelesaian secara sistematik. Dapatan kertas penyelidikan ini adalah untuk mendapatkan fungsi heuristik berharga secara automatik supaya keseluruhan proses penyelesaian masalah bebas dari pengetahuan manusia. Ini juga membantu menggunakan heuristic peraturan algoritma yang melakukan rawak berjalan kembali dari keadaan matlamat dan cuba untuk mengetahui sejauh mana mereka telah mencapai matlamat. Beberapa ujian dijalankan untuk membandingkan heuristik yang berbeza. Pertama, perbandingan dilakukan antara heuristik biasa (heuristic buatan manusia, model yang terlatih menggunakan peraturan heuristik), antara satu sama lain. Kemudian, perbandingan dilakukan antara heuristik yang dipelajari (model belajar dari heuristik biasa), antara satu sama lain. Akhirnya, membandingkan heuristik terbaik dari kumpulan pertama kepada kumpulan masing-masing dari kumpulan kedua. Hasilnya dianalisis sebagai prestasi heuristik yang berbeza adalah munasabah. Ini disebabkan oleh fakta bahawa bilangan nod diperluaskan untuk memasukkan algoritma A^*.

Kata kunci: Neural Networks, Rubik’s Cube, Pembelajaran Heuristik, Heuristik Admissible, Heuristik Inadmissible.

ACKNOWLEDGEMENTS

Primarily, I would like to thank my supervisor Dr.Aznul Qalid Md Sabri of Faculty of Computer Science and Information Technology in University Malaya, Malaysia. His clear guidance and advice have been nothing short of helpful.

I want to thank Miss Norhazariah Binti Husin, the administrative assistant in the Faculty of Computer Science and Information Technology in University Malaya, Malaysia Office. From the first day, we contacted University Malaya (UM), she has been there to assist us rain or shine through the year.

I want to thank my families, my partner and my friends, Mr. and Mrs. Magadevan Vasantha, Mr. Charkunnan Karuppanan, Dr.Milashni Sandran and Mr.Muhammad Nooraizad bin Zailani, for their unconditional love support, understanding and encouragement throughout the whole academic period from Degree to Master. Without them as my pillar of strength, it would have been impossible for me to be where I am today. Thank you with all my heart.

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF SYMBOLS AND ABBREVIATION

A^*	A Star
IDA^*	Iterative Deepening A Star
3D	3 Dimensions
ADI	Autodidactic Iteration
BFS	Breadth First Search
d	Scrambles
DNN	Deep Neural Network
GPU	Graphical Processing Unit
IT	Information Technology
LSTM	Long Short Term Memory
MCTS	Monte Carlo Tree Search
NP-Hard	Nondeterministic Polynomial Hard
PDB	Pattern Database

CHAPTER 1: INTRODUCTION

1.1 Research Background

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

Recently, they also succeeded in several purely combinatorial domains like the game of AlphaGo. Nowadays, the program AlphaGo (Silver, D, et al., 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 (Moravčík, et al., 2017) and several attempts have been made to use them for solving instances of Travelling Salesman Problem and other combinatorial problems (Bello, I. et al., 2017).

In development, 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 fine-tuned for specific problems.

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) (Morales et al., 2016), Deep Convolutional Networks, and Neural Turing Machines (Samadi, M et al., 2008).

1.2 Research 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. (Troy Fine and 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.

1.3 Research Problem

- Implementing neural network approaches to solve the Rubik’s Cube have struggled to succeed without human help and have had to rely on hand-engineered features and group theory to systematically find solutions.

1.4 Research Question

- How to obtain a valuable function automatically, so that the whole problem-solving process is free of human knowledge.
- How to use the rules of an algorithm that performs from a goal state and try to learn from the goal how far it can reach the rules.

1.5 Research Objective

- To obtain valuable heuristic functions automatically, so that the whole problem-solving process is free of human knowledge.
- To use heuristic rules of the algorithm to perform random walks back from the goal state and try to learn how far the goal has been achieved.

1.6 Research Significance

- The finding of this research paper is to obtain valuable heuristic functions that the whole problem-solving process is free of human knowledge and as use heuristic algorithm rules that will perform random walks backwards from a goal state and try to learn how far they reached rules are from the goal.

1.7 Research Delivery

- To prove this research paper and experiment of neural network using heuristic learning is the best choice to solve Rubik’s Cube.

1.8 Research Organization

This thesis consists of five chapters. Firstly, Chapter 1 will discuss the overview of Rubik’s Cube implementation on Neural Network, the motivation of the research, research problem, research objective, research question, research delivery, and research significance. Continually, the next chapter is literature review. In this section, will discuss the overview of Rubik’s Cube, and the algorithm used in other research papers (Pattern Database Learning, Boosting Learning, Autodidactic Iteration Learning, and Heuristic Learning) while comparing all these algorithms. In Chapter 3, methodology discussion on Heuristic Learning, using Admissible Learning and Inadmissible Learning. Later in Chapter 4, result and discussion, whereby various tests to compare different heuristics rules. Finally, Chapter 5 will conclude the research objective through the hypothesis made from the result.

CHAPTER 2: LITERATURE REVIEW

This chapter consists of a literature review of Rubik's Cube history as an introduction. Then continually the machine learning methods implemented in other research papers. Firstly, Pattern Database (PDB) Learning – solving Rubik's Cube by a method put away as a query table and applied to permutation problems. Secondly, the improvised machine learning method from PDBs is Boosting Learning - by using a weak prediction rule, the boosting algorithm combines these weak rules into a single prediction rule which more accurate. Thirdly, Autodidactic Iteration Learning (ADI) – an advance method of machine learning, also known as self-teaching Iteration, trains neural system esteem and strategy work through an iterative procedure. Finally, Heuristic Learning – a machine learning method which is an experience-based technique that helps in problem-solving, learning, and discovery. Similarly, as ADI but heuristic learning is mainly used to rapidly come to an optimal solution. Additionally, two heuristic algorithm which Admissible and Inadmissible are defined together. (*refer to Figure 2.1: Roadmap of Literature Review.*)

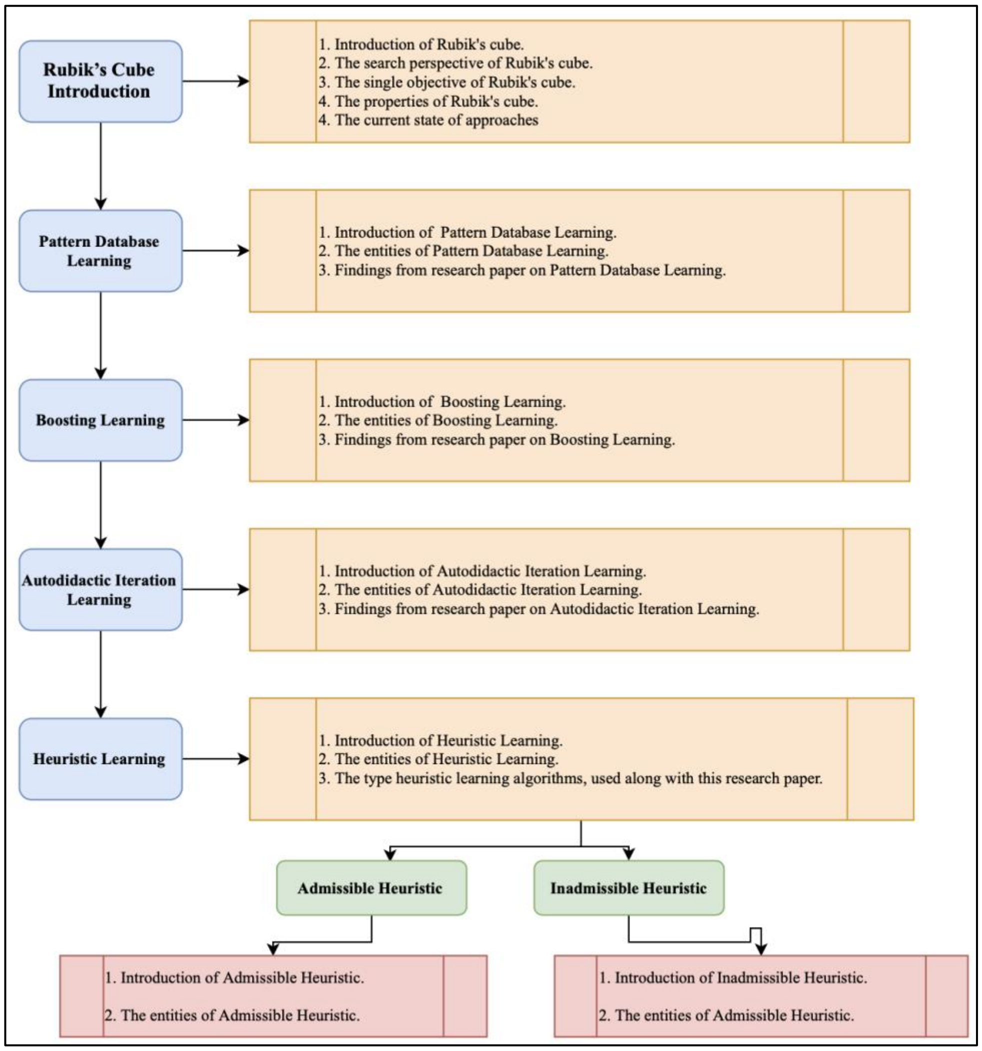

Rubik's Cube Introduction
1. Introduction of Rubik's cube.
2. The search perspective of Rubik's cube.
3. The single objective of Rubik's cube.
4. The properties of Rubik's cube.
4. The current state of approaches
Pattern Database Learning
1. Introduction of Pattern Database Learning.
2. The entities of Pattern Database Learning.
3. Findings from research paper on Pattern Database Learning.
Boosting Learning
1. Introduction of Boosting Learning.
2. The entities of Boosting Learning.
3. Findings from research paper on Boosting Learning.
Autodidactic Iteration Learning
1. Introduction of Autodidactic Iteration Learning.
2. The entities of Autodidactic Iteration Learning.
3. Findings from research paper on Autodidactic Iteration Learning.
Heuristic Learning
1. Introduction of Heuristic Learning.
2. The entities of Heuristic Learning.
3. The type heuristic learning algorithms, used along with this research paper.
Admissible Heuristic
Inadmissible Heuristic
1. Introduction of Admissible Heuristic.
2. The entities of Admissible Heuristic.
1. Introduction of Inadmissible Heuristic.
2. The entities of Admissible Heuristic.

Figure 2.1: Roadmap of Literature Review.

2.1 Rubik's Cube Introduction

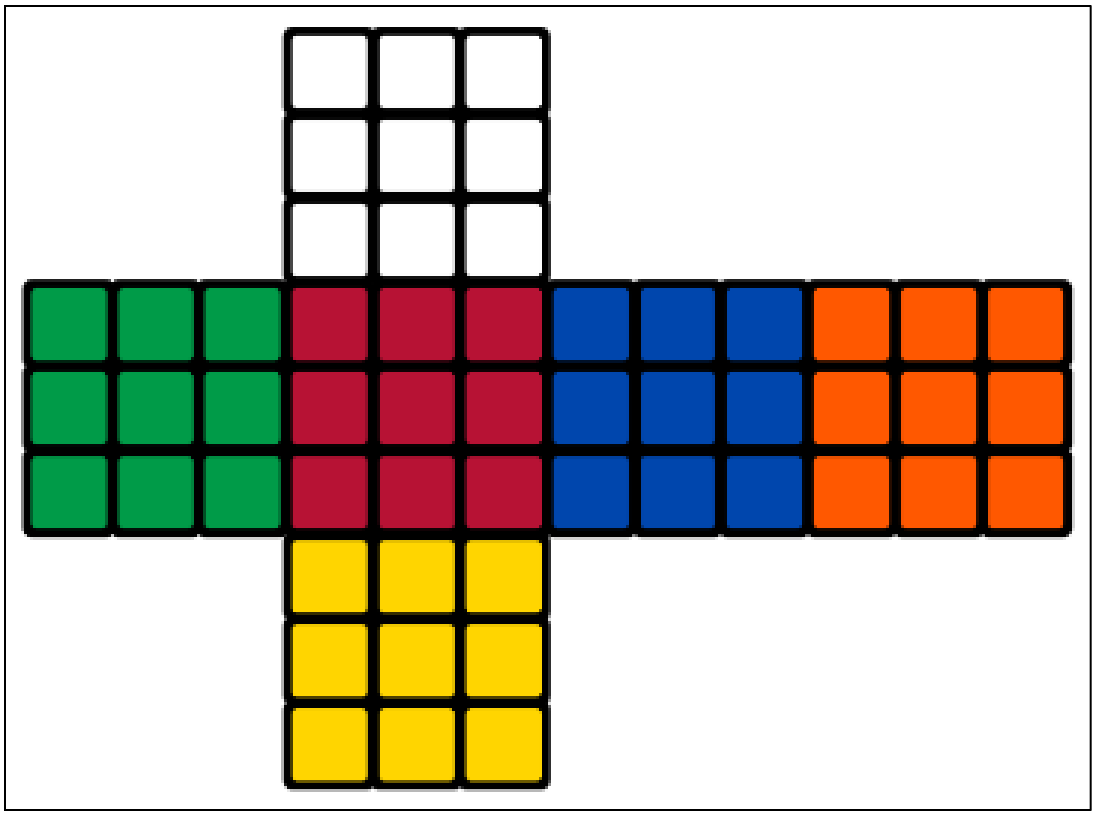


Figure 2.2: Rubik's Cube in 2 – Dimensional View.

Rubik's cube is a 3-Dimension (3D) puzzle, with 6 faces of different colour. It consists of $3^3 = 27$ cubes. From a search perspective, the number of different states of the standard 3 by 3 cube is $\mathbf{8! * 3^7 * \frac{12!}{2} * 12^{11}}$, which are over 43 quintillion legal positions of the Rubik's Cube, to be specific $\mathbf{43,252,003,274,489,856,000}$.

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 (*refer to Figure 2.2: Rubik's Cube in 2 – Dimensional View.)*. 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 $\emptyset(n^2)$, 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 (Demaine, E. et al., 2018). 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 (Sturtevant, N. R et al., 2014).

2.1.1 Pattern Database Learning

Pattern database (PDB) is a heuristic method put away as a query table and were applied to permutation problems. (Culberson J.C., 1996)

Initially, PDBs were proposed for solving single-agent problems. Today they are one of the most successful approaches for creating admissible heuristics. (Culberson J.C., 1996)

The arrangement of every single pattern database frames the area of the database. Given any permutation, the example can be gazed upward in the pattern database to locate the insignificant number of tasks required to put this N element in their right areas. A pattern database can be seen as a gathering of answers for sub goals that must be accomplished to solve the problems. (Alevizos – A – Bastas, 2017)

Even though using all PDBs and the reflection through the center symmetry gave the best results regarding the search effort the fastest combination was using only the corner, the last seven edge and the first seven edge PDBs. To summarize, there is a trade-off between the search effort and the node generation cost. The more the lookups, the lesser the search effort but, the more significant the cost in terms of time to generate a node. This is also where the power of locality preserving pattern databases lies. Neighboring values can be pruned without having to do a separate lookup, and this means that all searches required to generate the pruned nodes are translated into again in terms of time. (Alevizos – A – Bastas,2017)

An interesting observation from the research paper is that using the corner, and the two edge PDBs gave almost identical results in terms of search effort as using the corner and the last seven edge PDBs together with the reflection through the front face symmetry. Regarding dual lookups, they had the worst performance. This can be explained because the dual state's neighbouring values cannot be used to prune the normal state's neighbours. (Alevizos – A – Bastas, 2017)

2.1.2 Boosting Learning

Boosting, a machine-learning method that is the observation that finding many rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule. To apply the boosting approach, start with an algorithm for finding the uneven rules of thumb. This also called as a “weak” or “base” learning algorithm constantly, each time filling it a different subset of the training examples. Each time the algorithm is called, the base learning algorithm generates a new weak prediction rule, and after numerous rounds, the boosting algorithm must combine these weak rules into a single prediction rule that, hopefully, will be much more accurate than any one of the weak rules. (Robert E. Schapire)

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

As a conclusion from the resulting paper, there are some hypotheses for why boosting failed to improve performance. Assuming that boosting works better when weak learners are cheap to train and evaluate. This allows the researcher to generate an ensemble of many hundreds or thousands of learners, instead of tens. Additionally, weak learners should disagree on the optimal action when given the same input. Deep neural nets are costly to train, so do not have the time to train many weak learners - in testing, the researcher only had time to train 20 weak learners. Furthermore, because all gradient updates were applied to the same neural net, the neural net was regularized against changing too much each iteration. Upon manual inspection, the weak learners almost always agree on the optimal action (Alexander Irpan, 2016). Theoretically, a boosting approach will work with enough iterations, but boosting large neural nets appears to be less efficient than merely applying all data into a single neural net.

2.1.3 Autodidactic Iteration Learning

Autodidactic Iteration (ADI) or known as self-teaching Iteration, trains neural system esteem and strategy work through an iterative procedure. In each iteration, the inputs to the neural network are created by starting from the goal state and randomly taking actions.

The targets seek to estimate the optimal value function by performing a breadth-first search from each input state and using the current network to determine the value of each of the leaves in the tree. Updated value estimates for the root nodes are obtained by recursively backing up the benefits for each node using a max operator. The policy network is similarly trained by constructing targets from the move that maximizes the value. (McAleer, S. at el., 2018)

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 (v, p). 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 Monte Carlo Tree Search (MCTS) once the network is trained. Training samples for f 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. (McAleer, S. at el., 2018)

In the following research, (McAleer, S. at el., 2018), compares DeepCube against two different solvers. The primary standard is the Kociemba two-stage solver. This algorithm depends on human area learning of the gathering hypothesis of the Rubik's Cube. Kociemba will dependably comprehend any cube given to it, and it runs in all respects rapidly. In any case, in light of its broadly useful nature, it regularly finds an increasingly stretched out arrangement contrasted with different solvers. The distinctive standard is the Korf Iterative Deepening A* (IDA*) with an example database heuristic. Korf's algorithm will dependably locate the ideal arrangement from some random beginning state; be that as it may since it is a heuristic tree search, it will frequently need to investigate a wide range of states, and it will set aside an extended effort to process an answer. The specialist additionally looks at the full DeepCube solver against two variations of itself. To start with, they don't figure the most limited way of our hunt tree and rather extricate the underlying way from the MCTS: this will be named Naive DeepCube. Additionally, utilize the prepared esteem organize as a heuristic in a voracious best-first scan for a straight-forward assessment of the esteem arrange: this will be named Greedy. Subsequently, the researcher could not include Korf in the past examination since its runtime is restrictively moderate Note that Korf has one anomaly that is over 15 moves. This is on the grounds that Korf depends on the half-turn metric while we are utilizing the quarter-turn metric.

2.1.4 Heuristic Learning

Heuristic - a word derived from the Greek language for ‘find’ or discover is an adjective experience-based technique that help in problem-solving, learning, and discovery. (Katazyna at el., 2006)

A heuristic learning is mainly used to rapidly judgments (of a "target attribute”) that is computationally come to a solution that is hoped to be close to the best complex, and they instead substitute a more easily calculated possible answer, or 'optimal solution". A heuristic learning is mainly used to rapidly come to a solution that is hoped to be close to the best possible answer, or 'optimal solution'. (Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

The term heuristic is utilized for algorithms which discover arrangements among every conceivable one, however they don't ensure that the best will be seen, in this manner they might be considered as around and not precise calculations. These algorithms usually find a solution close to the best one, and they find it fast and efficiently. Sometimes these algorithms can be specific, that is they actually find the best solution, but the algorithm is still called heuristic until this best solution is proven to be the best. The method utilized from a heuristic calculation is one of the conspicuous methods, such as greediness, yet to be agreeable and fast, the calculation overlooks or even smothers a portion of the issue's requests. (Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

Two types of heuristics algorithm are utilized, firstly, an admissible heuristic or informed search is utilized to assess the expense of achieving the objective state in the heuristic algorithm. All together for a heuristic to be permissible to heuristic, the assessed cost should dependably be lower than or equivalent to the actual cost of achieving the objective state. The heuristic algorithm utilizes the admissible heuristic to discover an expected ideal way to the objective state from the current state. While an inadmissible which means that it might sometimes find the non-optimal path. However, it is also more efficient. By biasing the estimate, it can choose between path cost and search cost. (Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

2.1.5 Comparison of Learning

Different machine learning methods that can be used to solve the cube problem of the Rubik have been studied based on the literature review. To conclude, heuristics learning is the most appropriate approach and has the potential for improvement. (Table 2.1: Learning comparison)

As concluded, this research paper will implement two types of heuristics algorithms. Initially, the admissible heuristic to discover from the current state (the Misplaced Edges Cubes in Face and the Misplaced Corners Cubes in Face) is an ideal way to the objective state (the solved cubes in the face based on the color). While an inadmissible, it may sometimes find the non-optimal path. It's also more efficient though. It can choose between the cost of the path and the cost of the search by biasing the estimate. For example, finding the objective state (the color-based solved cube in the face) from the current state (the Misplaced Edges and Corner Cubes in Face).

Table 2.1: Comparison of Learnings

Learning	**Differences**
Pattern Database Learning	PDB is a heuristic method put away as a query table and were applied to permutation problems.
Boosting Learning	Boosting, a machine-learning method that is the observation that finding many rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule.
Autodidactic Iteration	ADI trains neural system esteem and strategy work through an iterative procedure.
Heuristic Learning	Heuristic learning s a solution that is hoped to be close to the best possible answer, or 'optimal solution'

CHAPTER 3: RESEARCH METHODOLOGY

This chapter consists of research methodology used along this research paper, which are Quantitative Research, Informed Search Algorithm and Learning the trained model. (*refer to Figure 3.1: Roadmap of Research Methodology*.)

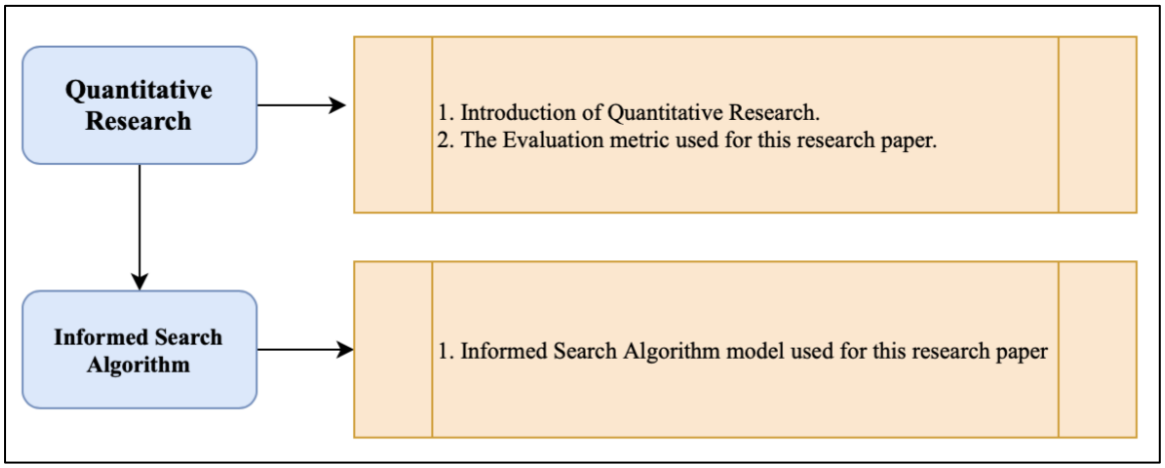

Quantitative Research
1. Introduction of Quantitative Research.
2. The Evaluation metric used for this research paper.
Informed Search Algorithm
1. Informed Search Algorithm model used for this research paper

Figure 3.1: Roadmap of Research Methodology.

3.1 Quantitative Research

Quantitative research techniques are examined 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 fact (Paul D. Leedy et al. 2018).

Quantitative is applied in this research paper, where one way to accomplish the research objective is to exploit statistical regularities of the problems.

3.2 Informed Search Algorithm

In this research paper, an informed search A^* with different heuristic used and as well machine learning estimator. The informed search A^* heuristics are described below (*refer to Figure 3.2: Flowchart of Informed Search Algorithm*).

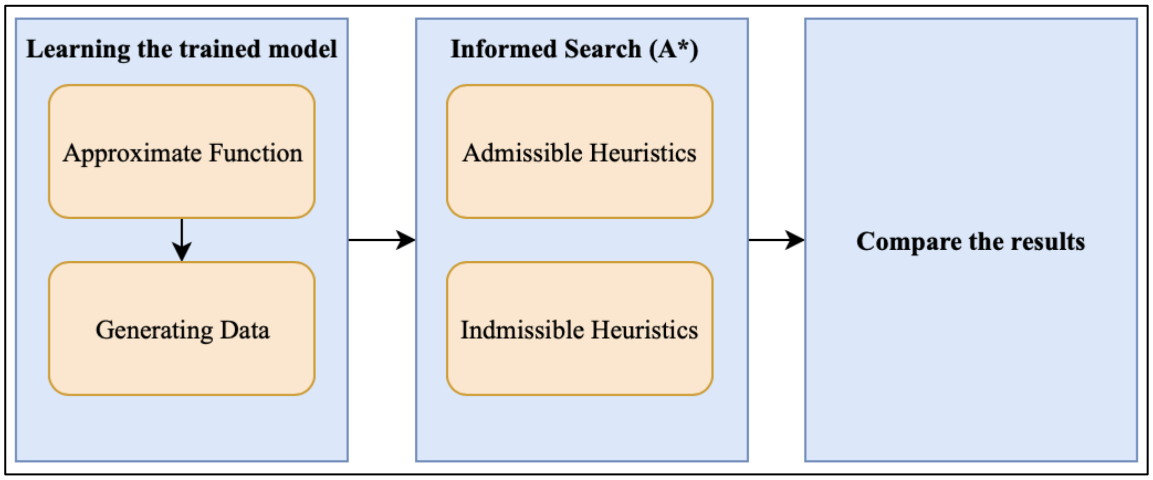


Figure 3.2: Flowchart of Informed Search Algorithm.

Afore the informed search A^* with different heuristic is implemented, a learning method in initiates before. Learning the trained model is used as an approximate function via Deep Neural Networks to train data an generates the trained data by allocating the size of the scramble. Later the Informed Search Algorithm A^* different heuristic using the trained data, the admissible and inadmissible heuristic ruled are applied. Finally, from the admissible and inadmissible rules applies, the results are compared.

3.2.1 Learning the trained models

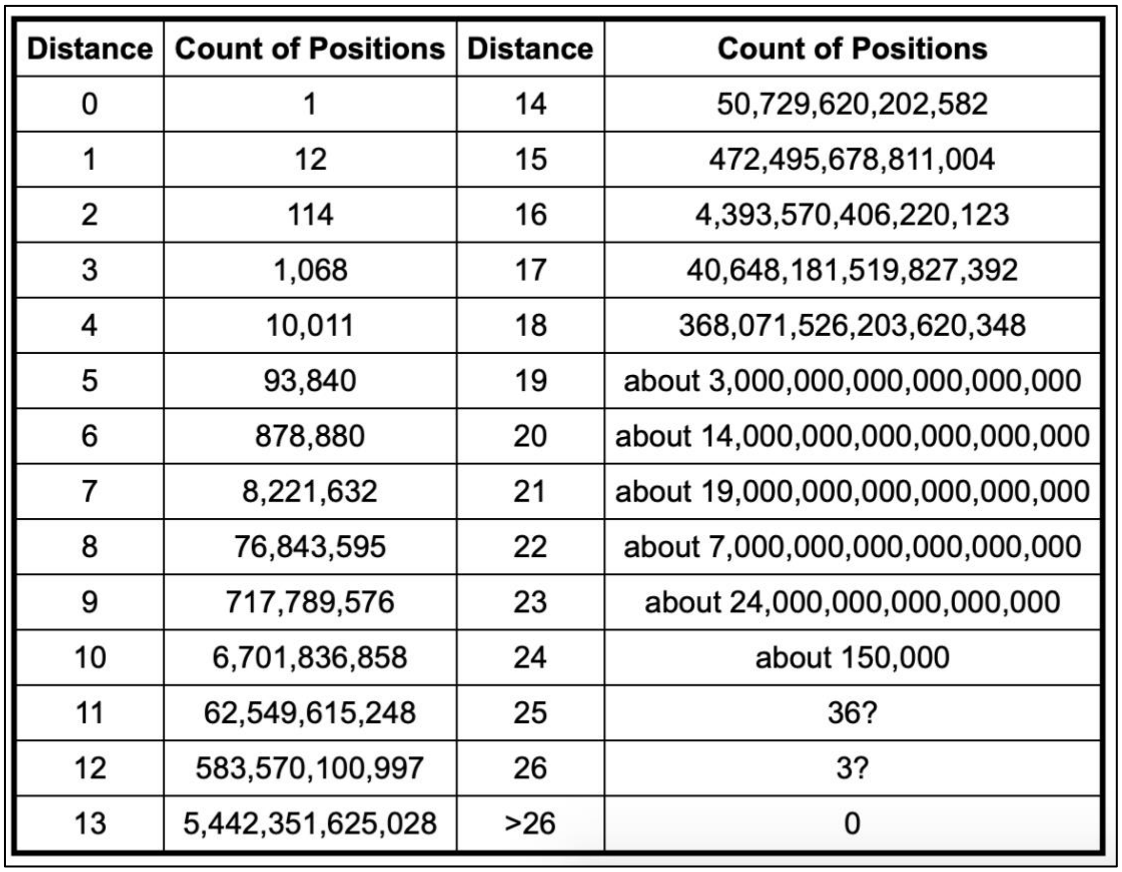
For this part, the trained model be used as an approximate function to learn, by using the Deep Neural. The steps explained as below.

3.2.1.1 Approximate Function

Let C be the space of all 3 by 3 of Rubik's Cube. The aim is to train a model that works closely with $h^{*}: C \rightarrow \mathbb{N}$ 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 (DNN) in a supervised learning setup.

3.2.1.1.1 Deep Neural Networks

It is known that any other continuous function over compact subsets of $\mathbb{R}^{n}$ can be combined with arbitrary precision using a neural feed network (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.



Distance	Count of Positions	Distance	Count of Positions
0	1	14	50,729,620,202,582
1	12	15	472,495,678,811,004
2	114	16	4,393,570,406,220,123
3	1,068	17	40,648,181,519,827,392
4	10,011	18	368,071,526,203,620,348
5	93,840	19	about 3,000,000,000,000,000,000
6	878,880	20	about 14,000,000,000,000,000,000
7	8,221,632	21	about 19,000,000,000,000,000,000
8	76,843,595	22	about 7,000,000,000,000,000,000
9	717,789,576	23	about 24,000,000,000,000,000
10	6,701,836,858	24	about 150,000
11	62,549,615,248	25	36?
12	583,570,100,997	26	3?
13	5,442,351,625,028	>26	0

Figure 3.3: Histogram of cube configurations (http://cube20.org/qtm/).

3.2.1.2 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* $d \approx 20$*, refer to Figure 3.3: Histogram of cube configurations (http://cube20.org/qtm/).),* 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.

Table 3.1: Average solution length, taken over 100 examples for each scramble (d).

d	7	8	9
$\bar{h}_3$	7.02	8	9
$\bar{h}_4$	7.02	8	9.14
$\bar{h}_5$	7.02	8	9

Used three methods to remove these cases:

1. (Trivial) Keep a set of visited states and make sure that do not repeated, and already visited state during the walk.
2. Three or more identical movements disallow. The sequence F, B, F, B, F' for example is limited to F'.
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 Sorting to exchange inappropriate actions. Finally, used the previous steps to reduce redundant measures after sorting the sequence.

3.2.1.2.1 Allocation for Choosing the scrambles (d)

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 d.

As the number of configurations grows exponentially, it does not make much sense to uniformly choose d from a set $\{\ 0,1,\ldots,d_{max}\}$. In fact, the learned model was not performable on larger distances after trying to train this way.

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

3.3.1 Admissible Heuristics

For this research paper, two admissible heuristic rules are used, Maximum Misplaced Edge Cubes in Face of Rubik's Cube and Minimum Misplaced Edge Cubes in Rubik's Cube Face.

3.3.1.1 Maximum Misplaced Edges Cubes in Rubik's Cube Face

Maximum Misplaced Edge Cubes in Rubik's Cube Face is one of the admissible heuristic rules. In this rule, which finding the maximum misplaced edges of the misplaced edges of cubes from the original centerpiece of the cube.

Each face of Rubik's cube, more than one number of wrong edge cubes will take place. For an example, the white face of Rubik's cube, will have maximum of 4 misplaced of edges cube (*refer to Figure 3.4: Cube notations for Misplaced Edges Cube*).

This pattern is acceptable as admissible rule, 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 of machine learning term:

$$h_1(c) = \frac{maximum}{f \in Faces\ (c)}\ \#misplaced - edges\ (f)$$

Equation 3.1: Maximum Misplaced Edge Cubes in Face.

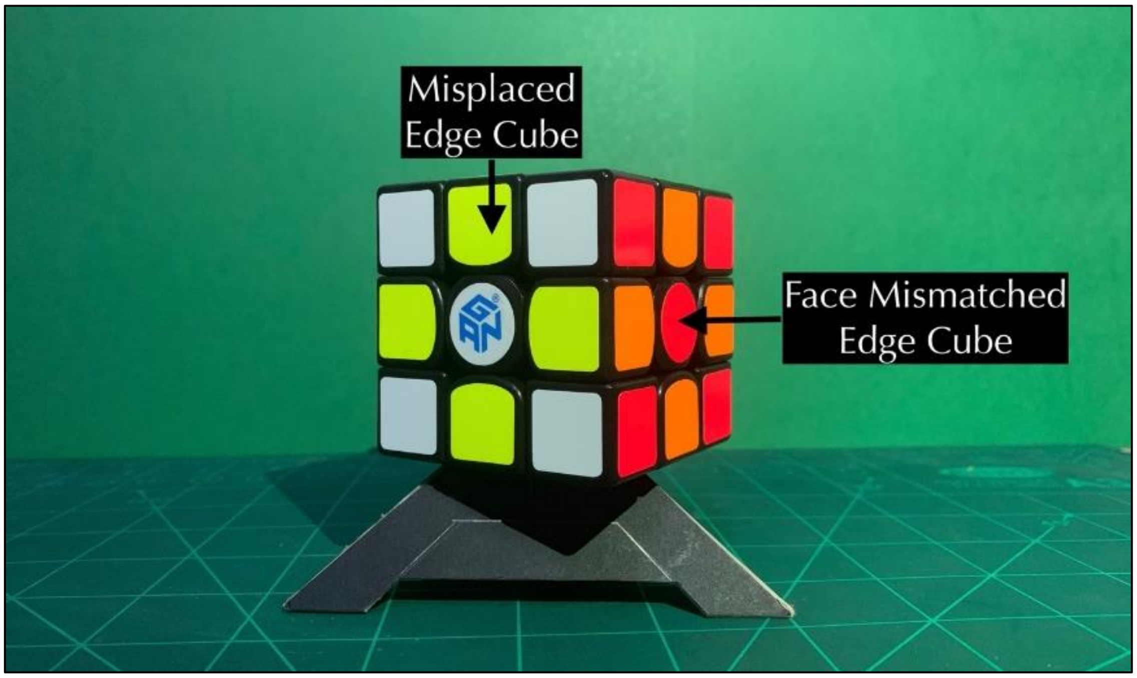


Figure 3.4: Cube notations for Misplaced Edges Cube.

3.3.1.2 Maximum Misplaced Corners Cubes in Face

Maximum Misplaced Corner Cubes in Rubik's Cube Face is one of the admissible heuristic rules. In this rule, which finding the maximum misplaced corner of the misplaced corner of cubes from the original centerpiece of the cube.

Each face of Rubik's cube, more than one number of wrong edge cubes will take place. For an example, the white face of Rubik's cube, will have maximum of 4 misplaced of edges cube (*refer to Figure 3.5: Cube notations for Misplaced Corners Cube*).

This pattern is acceptable as admissible rule, 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 of machine learning term:

$$h_2(c) = \frac{maximum}{f \in Faces\ (c)}\ \#misplaced - corners\ (f)$$

Equation 3.2: Maximum Misplaced Corners Cubes in Face.

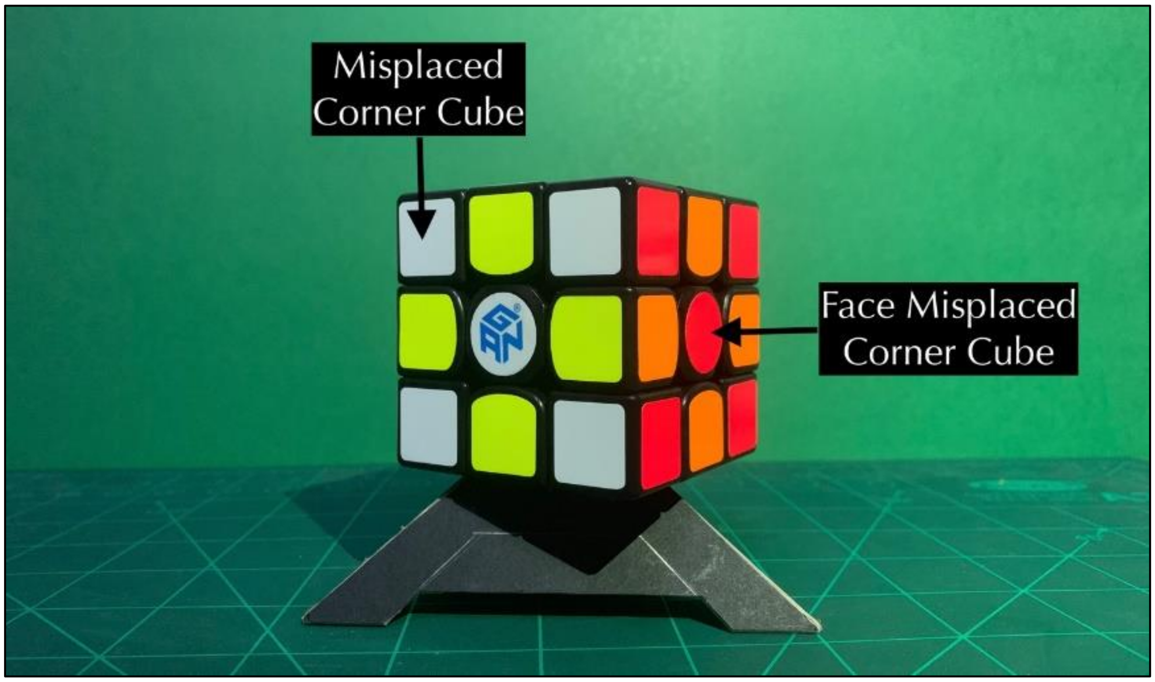


Figure 3.5: Cube notations for Misplaced Corners Cube.

3.3.2 Inadmissible Heuristics

For this research paper three admissible heuristic rules are used by finding the non-optimal path, Maximum Misplaced Edges and Corner Cubes in Face of Rubik's Cube, Maximum and Minimum Misplaced Corner and Edge Cubes in Face of Rubik's Cube and Maximum Misplaced Corner and Maximum Misplaced Edge Cube in Face of Rubik's Cube.

3.3.2.1 Maximum Misplaced Edges and Corner Cubes in Face

Maximum Misplaced Edges and Corner Cubes in Rubik's Cube Face is one of the inadmissible heuristic rules since this rule is bias-ly estimating the misplaced edges and corner. In this rule, which finding the maximum misplaced corner of the misplaced edges and corner of cubes from the original centrepiece of the cube.

Each face of Rubik's cube, more than one number of the wrong edge and corner cubes will take place. For example, the white face of Rubik's cube, will have greater than 0 for an unsolved cube piece and limited to 8.

This pattern is acceptable as an inadmissible rule because, in a solved cube, the edge and corner cubes have to be the same colour as the centre cube to 'fix', in the non-optimal path. In mathematical notation of machine learning term:

$$\bar{h}_3(c) = \frac{maximum}{f \in Faces\ (c)}\ [\#misplaced - edges\ (f) + \#misplaced - corners\ (f)]$$

Equation 3.3: Maximum Misplaced Edge and Corner Cubes in Face.

3.3.2.2 Maximum and Minimum Misplaced Corner and Edge Cubes in Face

Maximum and Minimum Misplaced Edges and Corner Cubes in Rubik's Cube Face is one of the inadmissible heuristic rules since this rule is bias-ly estimating the misplaced edges and corner. In this rule, which finding the maximum and minimum misplaced corner of the misplaced edges and corner of cubes from the original centrepiece of the cube.

Each face of Rubik's cube, more than one number of the wrong edge and corner cubes will take place. For example, the white face of Rubik's cube, will have greater than 0 for an unsolved cube piece and limited to 16.

This pattern is acceptable as an inadmissible rule because, in a solved cube, the edge and corner cubes have to be the same colour as the centre cube to 'fix', in the non-optimal path. In mathematical notation of machine learning term:

$$\bar{h}_4(c) = \frac{maximum}{f \in Faces\ (c)}\ g(f) + \frac{minimum}{f \in Faces\ (c)}\ g(f)$$

Equation 3.4: Maximum and Minimum Corner and Edge Cubes in Face.

3.3.2.3 Maximum Misplaced Corner and Maximum Misplaced Edge Cube in Face

Maximum Misplaced Corner and Maximum Misplaced Edges Cubes in Rubik's Cube Face is one of the inadmissible heuristic rules since this rule is bias-ly estimating the misplaced edges and corner. In this rule, which finding the maximum misplaced corner of the misplaced edges and corner of cubes from the original centrepiece of the cube.

Each face of Rubik’s cube, more than one number of the wrong edge and corner cubes will take place. For example, the white face of Rubik’s cube, will have greater than 0 for an unsolved cube piece and limited to 8.

This pattern is acceptable as an inadmissible rule because, in a solved cube, the edge and corner cubes have to be the same colour as the centre cube to ‘fix’, in the non-optimal path. In mathematical notation of machine learning term:

$$\bar{h}_5(c) = h_1(c) + h_2(c)$$

Equation 3.5: Maximum Misplaced Corner and Maximum Misplaced Cube Cubes in Face.

CHAPTER 4: RESULT AND DISCUSSION

In this research paper, several tests are ran to compare different heuristics. Firstly, compare the regular heuristics (human-made heuristic, the trained model using heuristic rules), with each other.. Then, compare the learned heuristics (the learned model from regular heuristic), with each other. Finally, compare the best heuristic from the first group to the respective one from the second group. The results are analyzed as a number of nodes expanded to include the A^* algorithm during operation is reasonable in the performance of various heuristics.

4.1 Python Scripts

The scripts are split into two main parts, which provided by the author of "*Solving the Rubik's Cube Without Human Knowledge, 2018*", for benchmarking. (Forest Agostinelli, 2018).

1. **The cube module** – Consists of a simple class representing a cube, the Rubik's Cube problem written as an A^* Search Algorithms and some other useful functions.
   PyPI package link: https://pypi.org/project/rubikai/
2. **Jupyter Notebooks 3** – Virtually hosted on the Google Collaboratory, which contain two parts:
   I. Model training:
      https://drive.google.com/open?id=1fDSMxdsxiGHj7tAJlTtCsfPYl4PZP5p7
   II. Main notebook:
      https://drive.google.com/open?id=1pVlrv83zAFDMYCok2cHIKIJbf9kCS6XT

4.2 Results

4.2.1 Trained Model Using Deep Neural Network

Three models are trained for a DNN regression with the different layer and neuron numbers, over 25 scrambles:

h_1: has three layers, of 70, 60 and layer of 50 neurons, respectively.

h_2: has four layers and 50 neuron levels each layer.

h_3: 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 to Figure 4.1: 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 to Figure 4.2: Model Evaluation of Average over 1600 examples.),* the evaluation loss was compared to the heuristics (*refer to Figure 4.3: Result of Heuristics Comparison Just Learned of Average over 5 iterations*.).

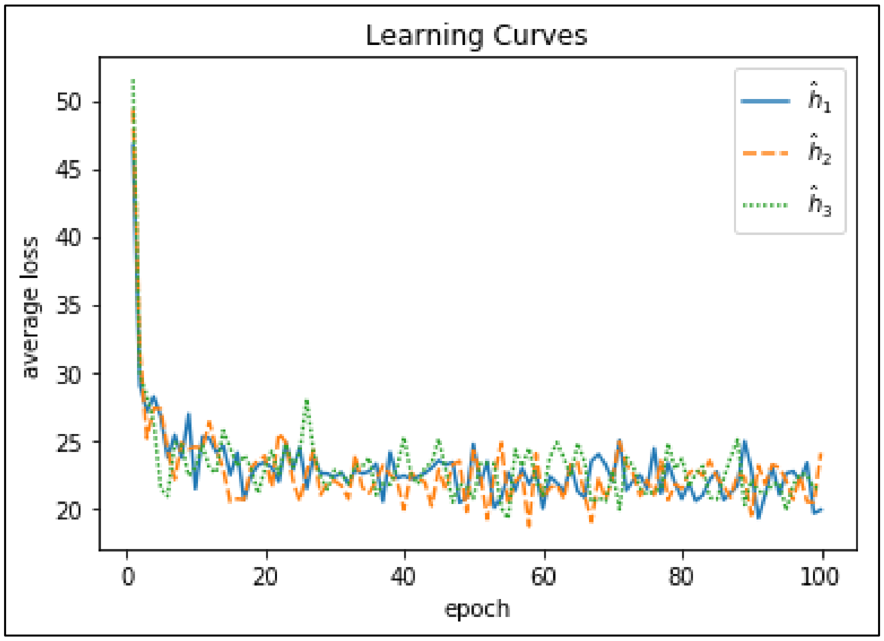


Figure 4.1: Learning Curves of Training Data for Plateau from 100 to 200 Epoch.

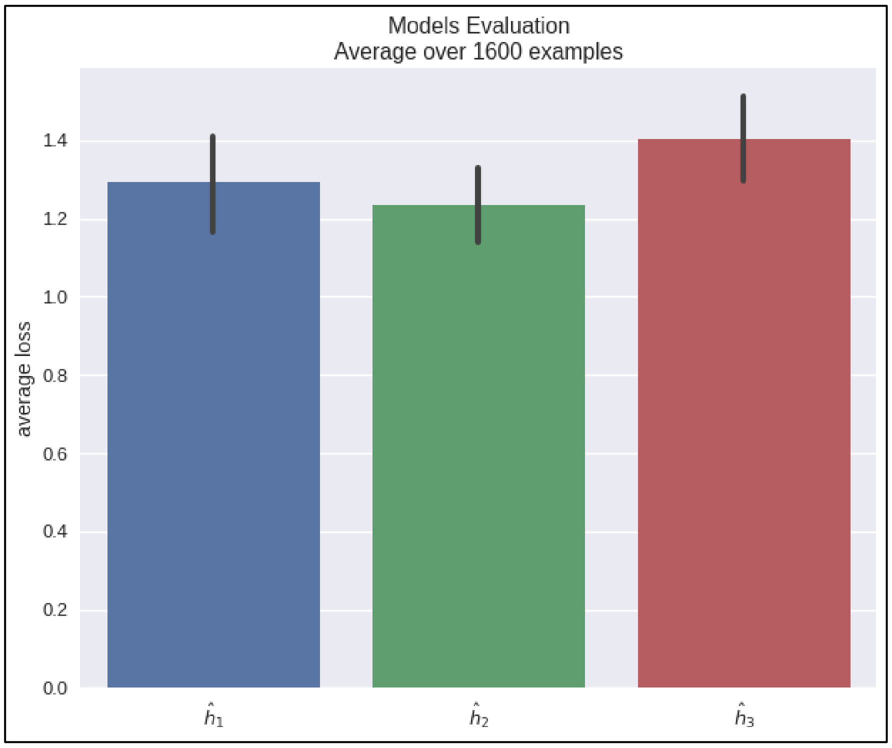


Figure 4.2: Model Evaluation of Average over 1600 examples.

4.2.2 Admissible Over Inadmissible.

As anticipated, the heuristics allowed for $d > 7$ (*refer to Table 4.3: Result of Heuristics Comparison Just Regulars of Average over 10 iterations.*) were too inefficient, so only used the inadmissible values for larger scrambles (d) values. These actions were successful in solving up to d = 25 moves.

The heuristic $\bar{h}_4$ has been especially successful with averages of 1, 500 nodes for 25 scramble movements from originally trained model (*refer to Figure 4.3: Result Heuristics Comparison Just Regulars of Average over 10 iterations*).

Furthermore, inadmissible 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 3.1: Average solution length, taken* *over 100 examples for each d.*, where for scramble = 1 to 6, the solutions returned have always been optimum.

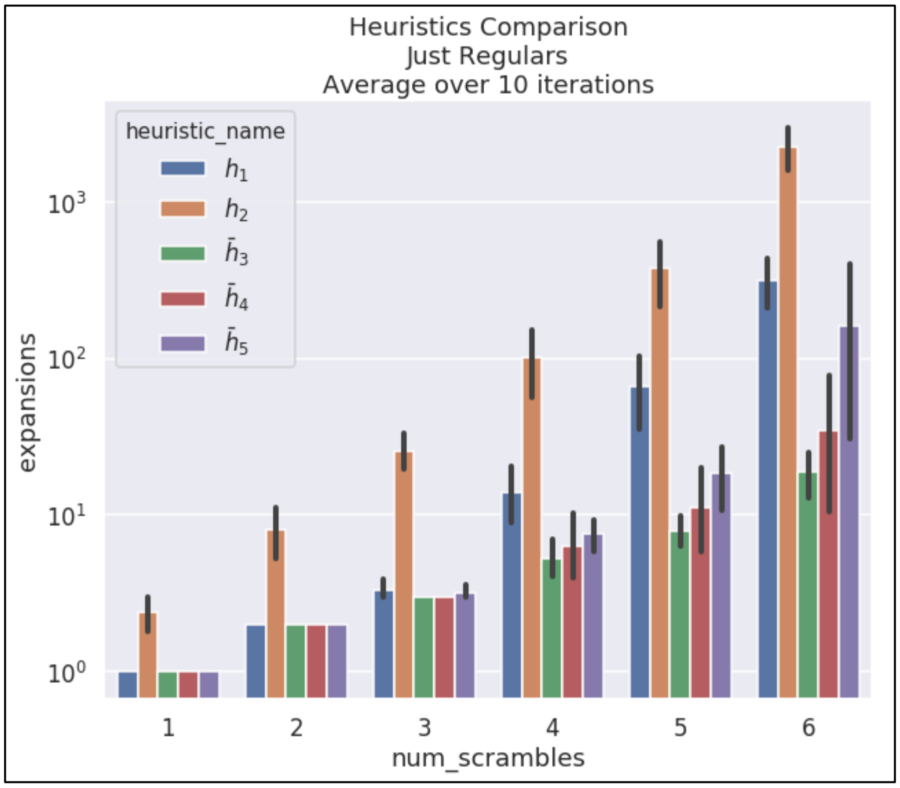


Figure 4.3: Result of Heuristics Comparison Just Regulars of Average over 10 iterations.

4.2.3 Learned Over Human-Made Heuristics

The learned heuristic, $\bar{h}_2$, is marginally better than the best person made, $\bar{h}_4$ as shown in the results (*refer to Figure 4.4: Result of Heuristics Comparison Regular Versus Learned of Average over 5 iterations.*).

During plotting the comparison, it was relatively accurate, and expected it to perform better. However, the results satisfied with the results given the limited training time and data.

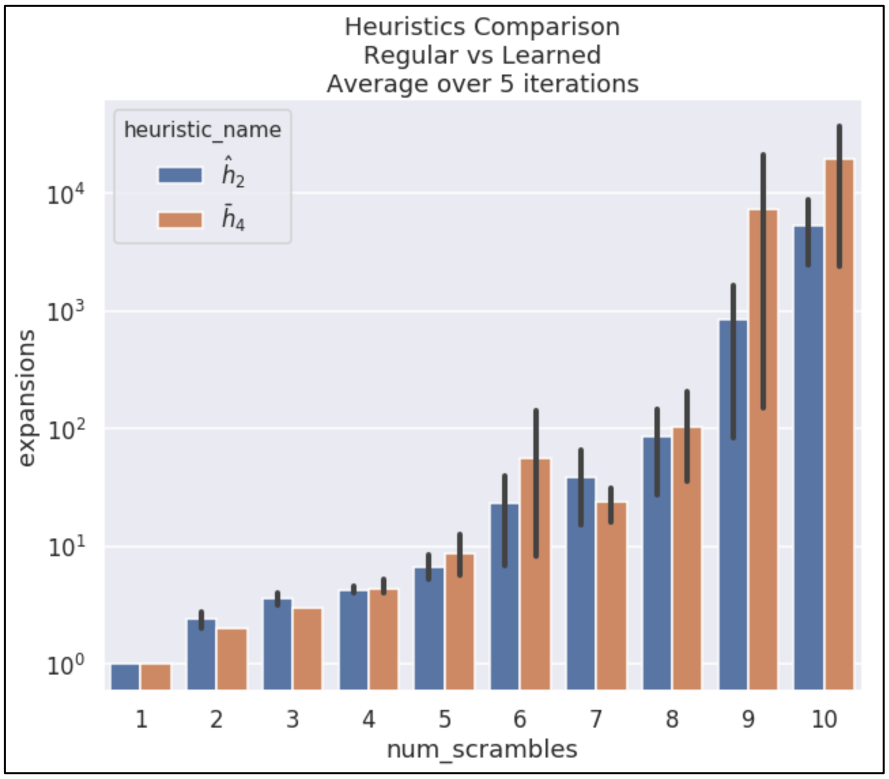


Figure 4.4: Result of Heuristics Comparison Regular Versus Learned of Average over 5 iterations.

CHAPTER 5: CONCLUSION AND FUTURE RESEARCH

5.1 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 three hours of training time per model, which is significantly less than other Rubik's Cube solvers. (Stephen McAleer, 2018)

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.

5.2 Limitation

- Required more time to train bigger size of epoch.
- Google Collaboration – GPU usage is close to limit, for executing the heuristic models.

5.3 Future Research

5.3.1 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.

5.3.2 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 (Harpreet Kaur, 2015), which can be used to improve the learned heuristic.

REFERENCE

Alevizos – A – Bastas. (2017). Pattern Databases for Search Problems: The Rubik's Cube Case. BSc Thesis. Total administration information and semiconduct department,national and capodistrian university of Athens

Alexander Irpan. (2016). Exploring Boosted Neural Nets for Rubik's Cube Solving. https://www.alexirpan.com/public/research/nips_2016.pdf

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2017). Neural Combinatorial Optimization with Reinforcement Learning. https://openreview.net/forum?id=Bk9mxlSFx

Csáji, B. C. (2001). Approximation with Artificial Neural Networks. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.2647&rep=rep1&type=pdf

Culberson J.C., Schaeffer J. (1996) Searching with pattern databases. In: McCalla G. (eds) Advances in Artificial Intelligence. AI 1996. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 1081. Springer, Berlin, Heidelberg.

Demaine, E., D, Eisenstat, S., & Rudoy, M. (2018, April 27). Solving the Rubik's Cube Optimally is NP-complete. Retrieved from https://arxiv.org/abs/1706.06708v2

Harpreet Kaur, (2015). Algorithms for solving the Rubik's cube. Retrieved from http://www.diva-portal.org/smash/get/diva2:816583/FULLTEXT01.pdf

Katazyna, J.K, and Jaszczolt, (2006). "Default in Semantics and Pragmatics" in Jaszzolt (ed) *The Stanford Encyclopedia of Philosophy.* Stanford: University Press

McAleer, S., Agostinelli, F., Shmakov, A., & Baldi, P. (2018). Solving the Rubik's Cube Without Human Knowledge. CoRR, abs/1805.07470. Retrieved from http://arxiv.org/abs/1805.07470.

Morales, F. J. O., & Roggen, D. (2016). Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. Sensors, 16(1), 115. Retrieved from https://doi.org/10.3390/s16010115

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., . . . Bowling, M. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science, 356(6337), 508-513. doi:10.1126/science.aam6960

Okechukwu Sunday Abonyi, Virginia Ogochukwu Ume, (2014). Effects of Heuristic Method of Teaching on Students' Achievement in Algebra. International Journal of Scientific & Engineering Research, Volume 5, Issue 2, February-2014. ISSN 2229-5518

Paul D. Leedy, Jeanne Ellis Ormrod, Laura Ruth Johnson. (2018), Practical Research Planning And Design. Twelfth Edition. https://www.pearsonhighered.com/assets/preface/0/1/3/4/0134775651.pdf

Samadi, M., Felner, A., & Schaeffer, J. (2008). Learning from Multiple Heuristics. http://www.aaai.org/Library/AAAI/2008/aaai08-056.php

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., . . . Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529, 484. Retrieved from https://doi.org/10.1038/nature16961. doi:10.1038/nature16961

Sturtevant, N. R., Felner, A., & Helmert, M. (2014). Exploiting the Rubik's Cube 12-Edge PDB by Combining Partial Pattern Databases and Bloom Filters. http://www.aaai.org/ocs/index.php/SOCS/SOCS14/paper/view/8942

Troy Fine & Boris Gorshenev, (2018). Progress Report Rubik’s Cube, CMPS 140