The most efficient algorithm to solve a Rubik's cube

Science Research project

Justin Marcellienus

10 Polding

The most efficient algorithm to solve a Rubik's cube

Aim

Constructing a Lego Rubik's cube solver (most efficient method of solving a Rubik's cube)

Introduction

The Rubik's Cube is a 3-D combination puzzle invented in 1974 by Hungarian sculptor and professor of architecture Ernő Rubik. Since then its immense success has led to it becoming the world's most successful toy in history with nearly 350 million units being sold worldwide. Despite the relatively simple concept, the cube has over 43 Quintillion (43,252,003,274,489,856,000) different combinations of scrambling. Nevertheless the legal arrangement of the Rubik's Cube can be solved in 20 moves or fewer, with the use of a variety of algorithms. The most important part of solving a Rubik's Cube is understanding how it works. When looking at a Rubik's Cube, there are six sides, each containing nine pieces. The sides can be rotated in many ways, but regardless of what is done to the cube (unless taken apart) the centre pieces don't move with respect to each other. Therefore, when the cube is being solved, the central pieces cannot move position.

The Rubik's can be solved using a range of different algorithms ranging from layered, which can be done by hand using patterns, or heuristic which require complex equations that subdivide a cube requiring connection to a PC for extra operating power.

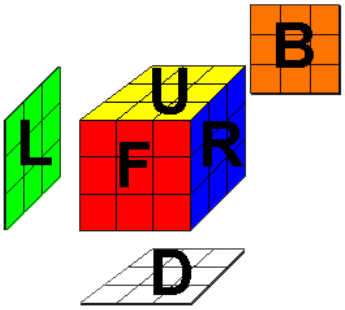
The problem that will be investigated is the construction of a Rubik's cube solver using Lego Mindstorms (robotics kit) and using software to test several different algorithms/methods of solving the cube. Each method is used to solve a standard 3x3 Rubik's cube to determine which algorithm would take the least number of moves within the least period of time. To understand the algorithms, the Rubik's cube is notated based on side, turns and cube rotation, to allow for simplified equations. To denote a sequence of moves on the 3×3×3 Rubik's Cube the "Singmaster notation" is applied which was originally proposed by David Singmaster in 1979.

Cube Notation (Singmaster notation)

Faces

There are 6 faces on a cube. Each face is represented by a letter, according to where it is located. These faces make the most sense when you hold the cube with one face parallel to the ground and one face facing you, but algorithm pages will often display the cube so that you can see the front, right, and top faces. The six faces are:

- **F** (Front) - the side facing you.
- **U** (Up) - the side facing upwards.
- **R** (Right) - the side facing to the right.
- **B** (Back) - the side facing away from you.
- **L** (Left) - the side facing to the left.
- **D** (Down) - the side facing downwards.



Turns

A turn of one layer of one of the six faces of the cube is written by adding a suffix (F, U, R, B, L, and D) to the face's name. There are three possible turns that can be applied to a face and all moves should be applied as if you were looking at the face straight-on. Using the U face as an example, the following are possible turns:

- **U** - A 90-degree clockwise turn of the U face.
- **U'** - A 90-degree counter clockwise turn of the U face.
- **U2** - A 180-degree turn (either clockwise or counter clockwise) of the U face.

Cube Rotations

Cube rotations involve turning the entire cube. Although it does not count as a “move” it helps change cube perspective to shorten algorithms. The possible cube rotations, which can also be modified with **'** (90 degree counter-clockwise) or **2** (180 degree turn clockwise or anti-clockwise) like a face turn are:

- **x** or **[r]** - a rotation of the entire cube as if doing an R turn.
- **y** or **[u]** - a rotation of the entire cube as if doing a U turn.
- **z** or **[f]** - a rotation of the entire cube as if doing an F turn.

Cube algorithms

Three popular algorithms exist for solving the cube – Thistlethwaite’s algorithm, Kociemba’s Algorithm and Korf’s Algorithm. Kociemba’s Algorithm was an improvement on Thistlethwaite’s algorithm. Korf’s Algorithm was developed by Richard Korf in 1997. He claimed to optimally solve the cube by iterative deepening. With his algorithm he claimed one could solve the cube in 18 moves.

Thistlethwaite's algorithm

Made by: Morwen Thistlethwaithe

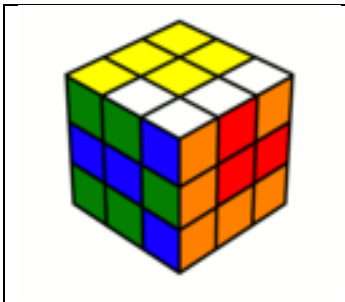
Date: 1981

Average moves: 45

The way the algorithm works is by restricting the positions of the cubes into groups of cube positions that can be solved using a certain set of moves.

Group	**Description**	**Formula**
Group 0	This group contains all possible positions of the Rubik's Cube	G_0 = <L,R,F,B,U,D>
Group 1	Positions that can be reached from the solved state with quarter turns of the left, right, front and back faces of the Rubik's Cube, but only double turns of the up and down sides.	G_1 = <L,R,F,B,U2,D2>
Group 2	Restricted to turns that can be reached with only double turns of the front, back, up and down faces and quarter turns of the left and right faces.	G_2 = <L,R,F2,B2,U2,D2>
Group 3	Positions in this group can be solved using only double turns on all sides.	G_3 = <L2,R2,F2,B2,U2,D2>
Group 4	The final stage, completely solved	G_4 = {I}

Kociemba's Algorithm



Made by: Herbert Kociemba

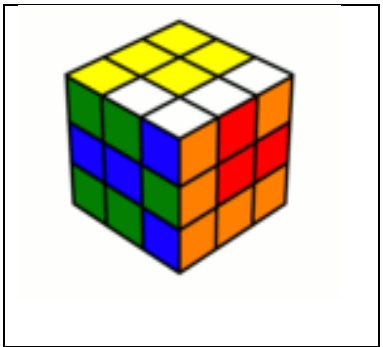
Date: 1992

Average moves: 20

Thistlethwaite's algorithm was improved by Herbert Kociemba in 1992. He reduced the number of groups to only two therefore making a substantial decrease in required moves to a maximum of 29 moves and a minimum of 19

Group	**Description**	**Formula**
Group 0	All possible positions of the cube	$G_0 = < L,R,F,B,U,D >$
Group 1	Split into the top half of the cube which uses the IDA formula to subdivide and solve	$G_1 = <U,D,L^2,R^2, F^2,B^2>$
Group 2	Split into the bottom half of the cube which uses the IDA formula to subdivide and solve	$G_2 = <L2,R2,F2,B2,U2,D2>$

Korf's Algorithm



Made by: Richard Korf

Date: 1997

Average moves: under 20

Korf's algorithm is based on the works of Kociemba's algorithm in terms of splitting the cube into subgroups. However he simplified it down to a mere 2 groups using the IDA* code. The IDA code is a general search algorithm that simplifies the steps required to travel from the root to the solution using a complex code called the Psuedocode. First he identified a number of sub problems that are small enough to be solved optimally:

1. The cube restricted to only the corners, not looking at the edges
2. The cube restricted to only 6 edges, not looking at the corners or at the other edges.
3. The cube restricted to the other 6 edges.

The Lego Robot

The LEGO Mindstorms NXT is a programmable robotics kit released by LEGO in late July 2006, it comes with:

- 1 NXT processor brick
- 3 servo motors
- 1 colour sensor
- 1 ultrasonic sensor
- 2 touch sensors

It comes with the NXT-G programming software, or LabVIEW for LEGO MINDSTORMS. A variety of unofficial coding languages exist, such as NXC, NBC, leJOS NXJ, and RobotC that can be read by the CPU block.

NXT Intelligent Brick

The main component in the kit is a brick-shaped computer called the NXT Intelligent Brick. It can take input from up to four sensors (2 touch sensors, ultrasonic sensor and colour sensor) and control up to three servo motors, using connecting RJ12 cables. The brick has a 100×60 pixel black and white LCD screen and four buttons that can be used to navigate user interface menus. It has a 32-bit ARM7TDMI-core Atmel AT91SAM7S256 microcontroller with 256KB of FLASH memory and 64KB of RAM, plus an 8-bit Atmel AVR ATmega48 microcontroller, and Bluetooth support. It also has a speaker and can play sound files at sampling rates up to 8 kHz. Power is supplied by 6 AA (1.5 V each) batteries in the consumer version of the kit.

A Rubik's cube solver will need to use colour sensors to detect the colours and transfer the data to the central NXT brick, where it is solved. Then the solution needs to be translated into actions for the servo motors to turn the cube and twist layers. Once the basic functions of the motors are programed, it should be relatively easy to swap out the programing with each algorithm.

The robot will have a flat base with a rotating turntable that will house the Rubik's Cube. It will include an arm to flip the cube by tilting it over the turntable and guiding it in place. Finally, a colour sensor will be mounted to scan the colours of each face and transfer information to the central brick were it will process the solution. During construction the robot is split into 4 main parts which are added together, these are the:

- Flipping arm
- Color sensor arm
- Turntable
- Robot Base

NXT programing Software

The NXT programming software that is bundled with the Mindstorms kit is a NXT-G v2.0 is a graphical programming environment that can be used for real-world programming. The coding language supports virtual instruments for all LEGO branded and most 3rd party sensors/components. Although it is rather basic, predominantly used for parallel sense and respond loops (e.g. wait 60 seconds and play a "beep"), it can be used in conjunction with a multitude of other coding software, opening it up to much more advanced functions.

Variables

Independent variable

The independent variable will be the algorithm used to determine the solution for the cube. The three algorithms tested will be:

- Thistlethwaite's
- Kociemba's
- Korf's

Each algorithm will need to be transferred to the NXT program and uploaded to the robot for each test.

Dependent variable

During the investigation the amount of time taken for the Rubik's cube to be completely solved will be measured starting from the scanning stage to the final completion stage. Additionally the number of moves taken to reach the solved state will be recorded.

Controlled variables

Variable	How could it affect your experiment?	How will it be controlled?
Design of the robot	Altering the design of the robot midway through the experiment could result in changes of efficiency, possibly leading to discrepancies in the time taken to solve the cube	This could be controlled by ensuring all of the pieces in the robot are the same for each test. A pre-test check should be completed before resuming the next test.
Battery life	The battery life of the 6 AA batteries used to run the robot can often run flat quickly, leading to a significant reduction in power to the servo motors making them run slower. This could lead to the results being inaccurate due to differences in the motor power	To alleviate the affects, 6 energizer AA rechargeable batteries will be used. They will be fully charged before each test
Jumbled position of the cube	The position/state of the cube must be identical in all test setups otherwise it may change the amount of moves required to solve the cube	Before each test, the cube must be jumbled in the identical state so it is the same for each test
Determining when to start/stop timer	Starting the timer at the right time is important as, failing to do so may result in the results being inaccurate	This issue can be alleviated by starting the timer at specific point, such as soon as the robot starts scanning and stopping it once the last move is completed.

Hypothesis

I believe that Richard Korf's algorithm will be the most efficient method to solve the cube. According to previous research, it is can be deduced that Korf's algorithm will require the least number of moves and time due to the fact that it elaborates on the findings of all previous formulae. Additionally it uses CPU power and RAM from a PC to calculate solutions to the subdivided algorithms in the shortest time.

Equipment

Equipment that is needed to do the experiment/s includes:

- 1 Complete Lego Mindstorms NXT kit
- Extra Lego pieces
- Rubik's Cube
- Each Algorithm code transferred to NXT program
- High powered computer with a minimum of 8gb RAM and an high end processor (over 3ghz)
- 6 Rechargeable AA batteries
- Stopwatch

Method

Robot construction

1. Construct the flipping arm that fits around the Rubik's cube face, using 1 servo motor and other pieces
2. Construct the Colour sensor arm, using 1 colour sensor and 1 servo motor. The colour sensor should be positioned so that it will be close to the Rubik's cube in operation (to improve colour detection)
3. Construct the turntable for the Rubik's cube, ensuring that the cube fits snugly with minimal space to move, however not completely jammed. Connect to 1 servo motor
4. Construct the base of the Robot, ensuring it is completely flat with no pieces obstructing the flipping arm or colour sensor.
5. Connect the RJ12 cables to the corresponding colour sensor and motor:

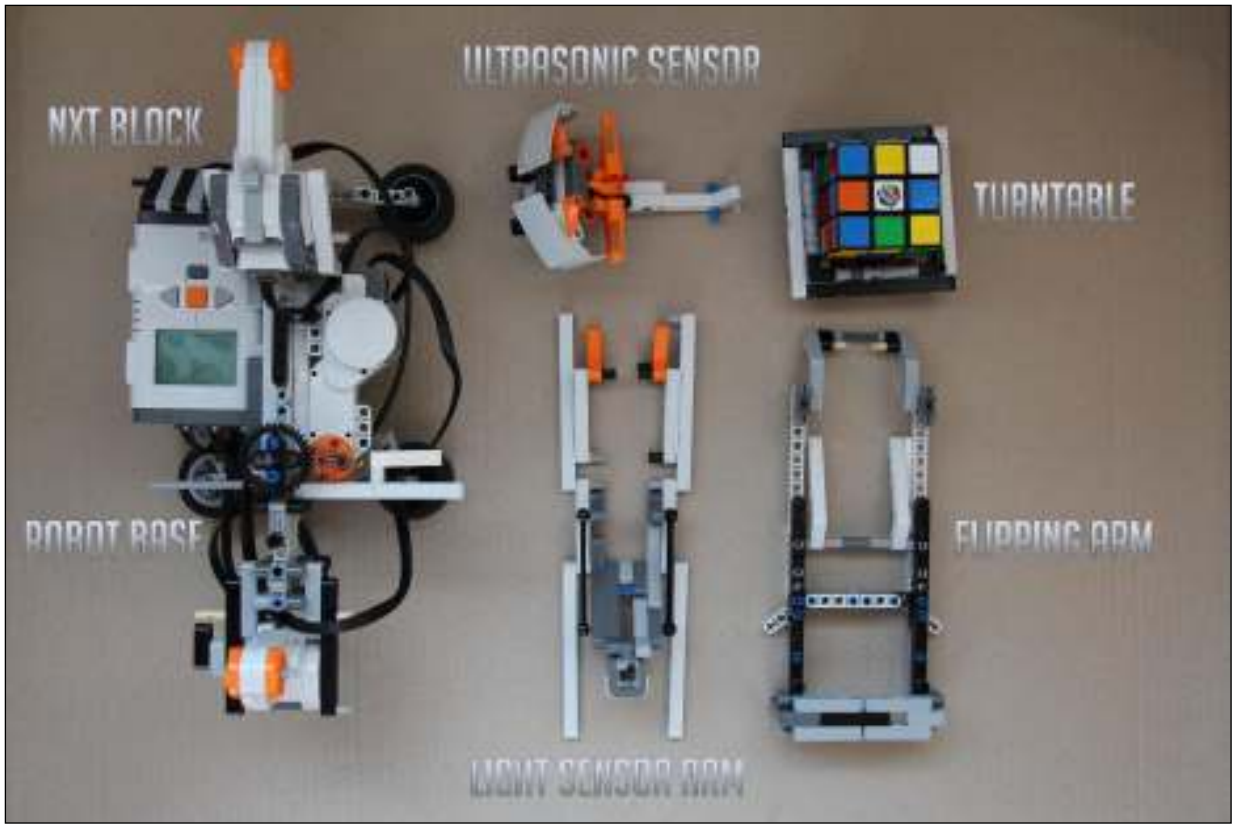
   1 - Color sensor

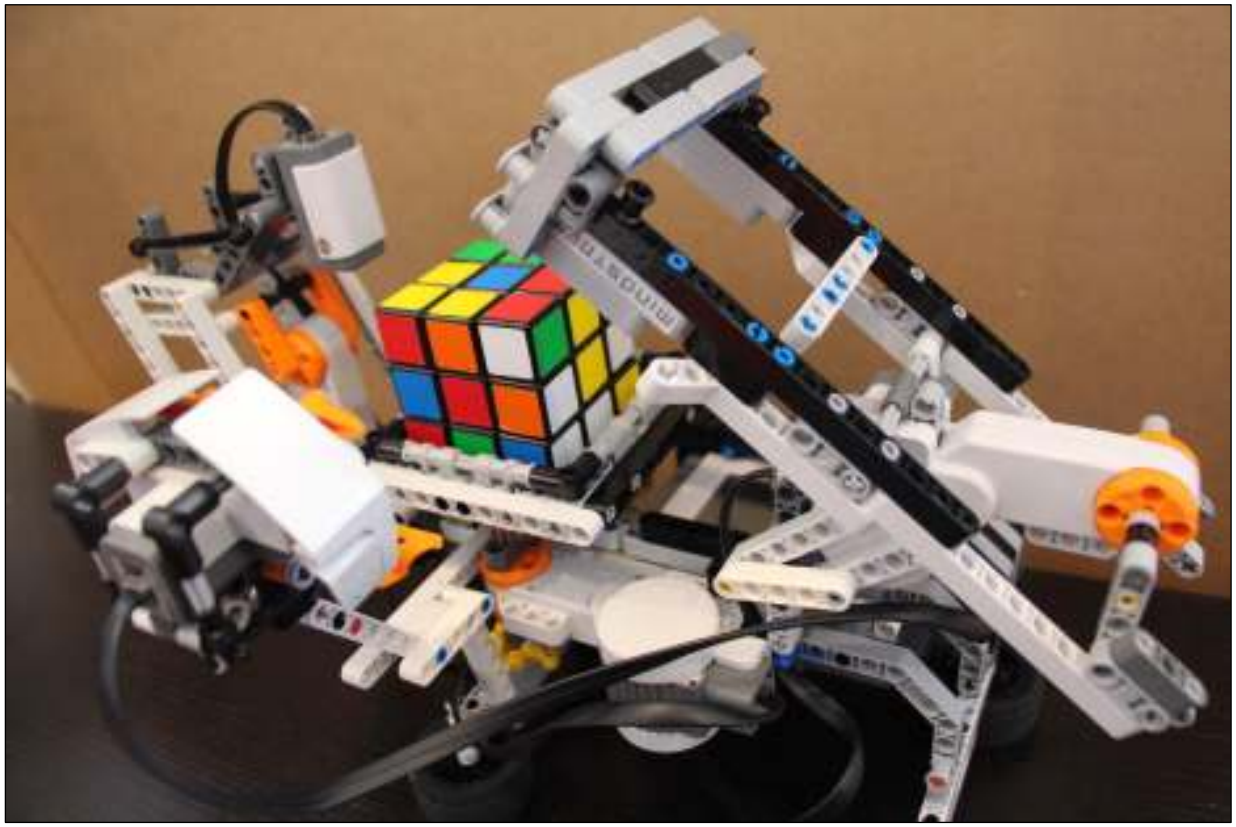
   2 - Ultrasonic sensor

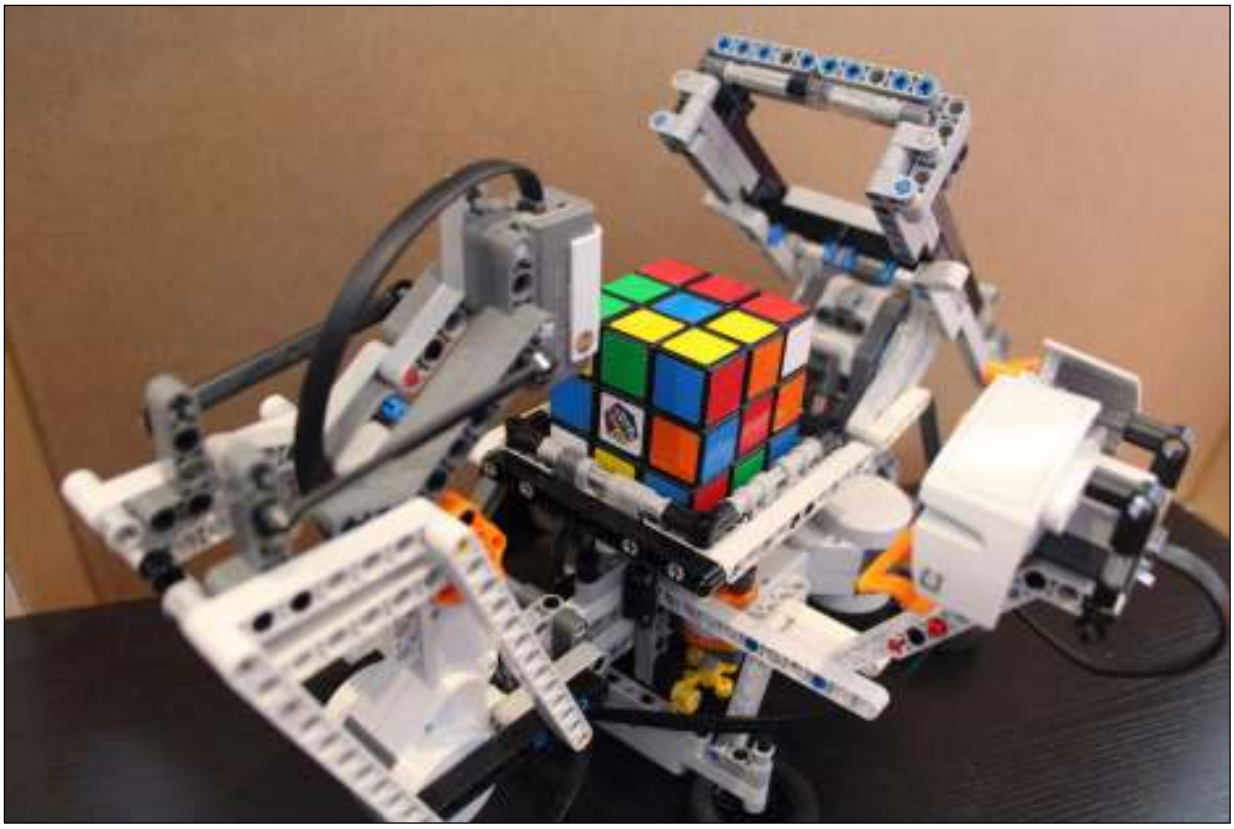
   A - Turntable motor

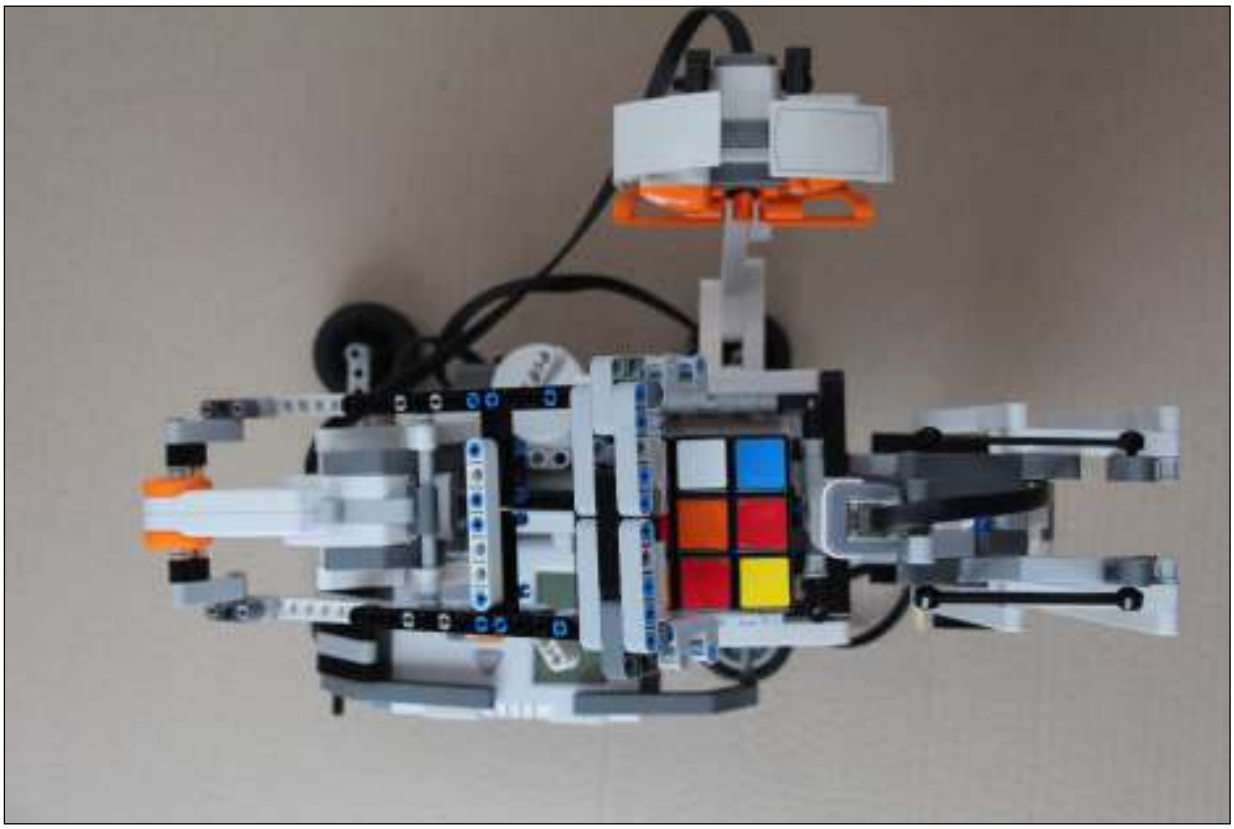
   B - Tilter arm motor

   C - Color sensor motor


ULTRASONIC SENSOR
NXT BLOCK
TURNTABLE
ROBOT BASE
FLIPPING ARM
LIGHT SENSOR ARM

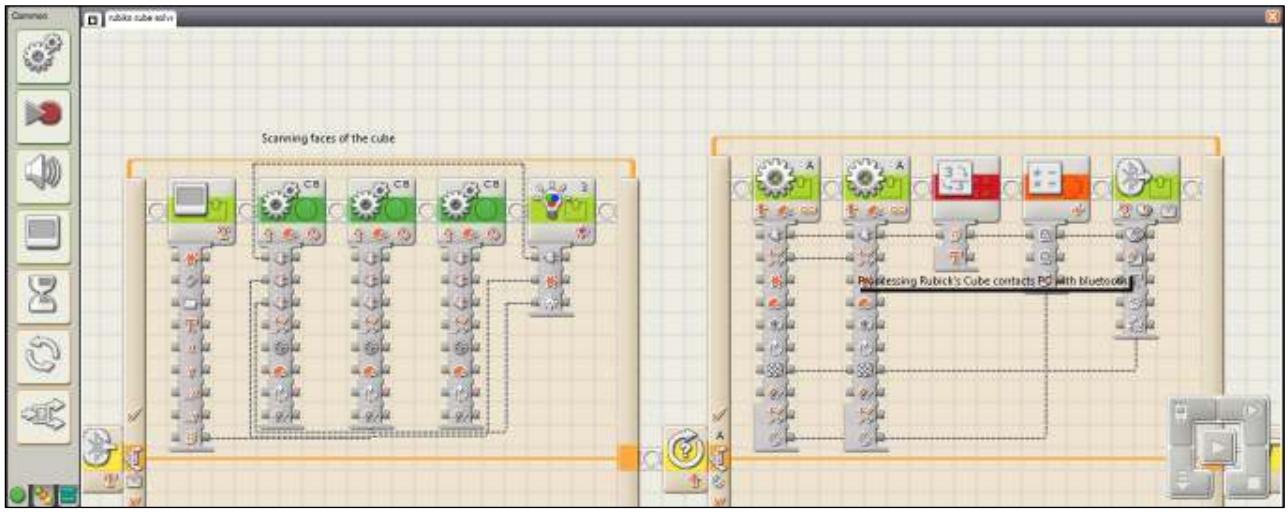


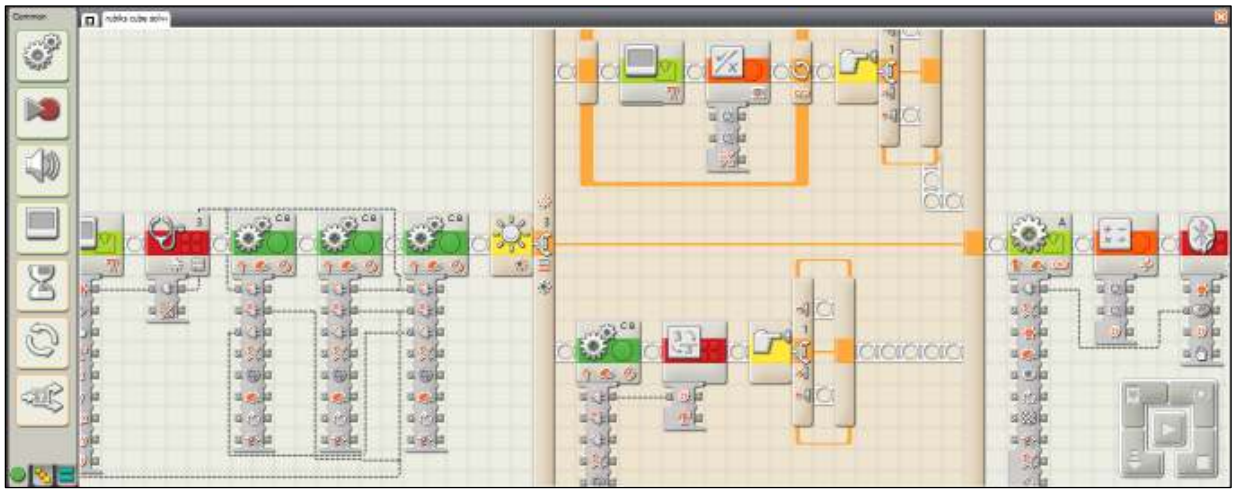




Robot coding

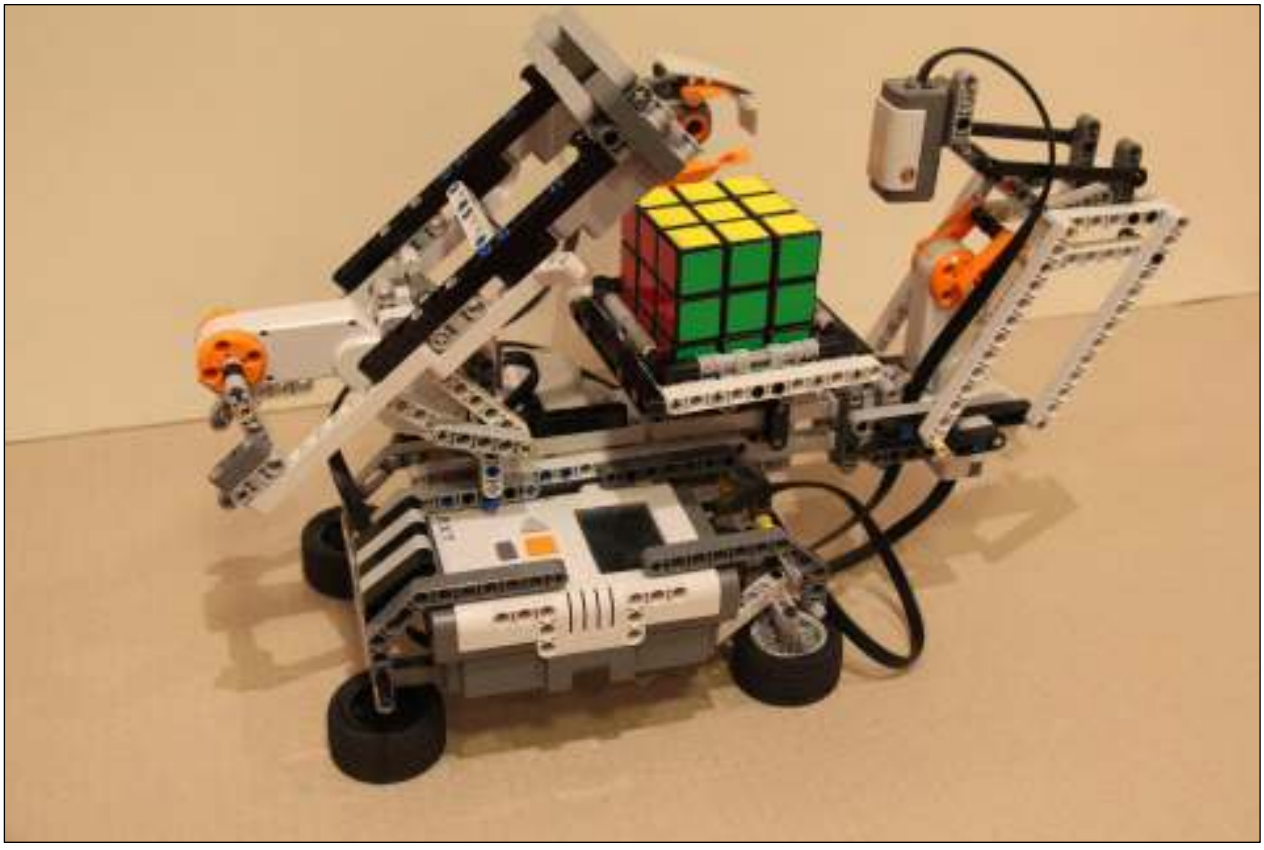
1. Using a simple algorithm, set the root program in NXT with each “string” of program for each particular move and create a template.
2. Download the Thistlewaite source code onto a computer with the NXT program installed
3. Using the NXT software and a java application, transfer the code to the NXT template made in step 1.
4. Ensure the code is compatible with the NXT and check for any “broken lines” in the code
5. Connect the NXT block to a computer via Bluetooth or cable
6. Upload the executable NXT file to the processing brick
7. Run the calibration test, to place motors in the right position
8. Repeat steps 2-7 for the Kochiemba and Korf Algorithms





Conducting the experiment

1. Ensure the 6 AA batteries are fully charged and inserted into the NXT brick
2. Scramble the brick into the exact pre-assigned position
3. Place the Rubik's cube onto the turntable
4. Turn on the Robot and launch the executable program file
5. Start the timer as soon as a "beep" sounds before the scanning
6. Record the amount of moves taken to solve the cube
7. Stop the timer on the second "beep"
8. Repeat the experiment on the same algorithm 3 times to ensure consistency
9. Repeat steps 1-7 for the Kochiemba and Korf algorithms
10. Record all results and arrange into a table



Risk assessment

Risk	How the risk will be reduced or avoided	Risk Level
Short circuit	This risk could be avoided by making sure wires are on the right connection and not being obstructed by anything before connecting the power.	Medium
Computer fire hazard	Due to the heavy workload on the computer components trying to solve algorithms, the computer must have adequate cooling to keep the hardware at a safe operational temperature (below 80 degrees)	Low
Fingers can get jammed in the motor	Do not place fingers near the robot's motors whilst in operation	Low

Results

After completing the experiment, the results reflect what I expected upon research on the internet. In summation, the results proved:

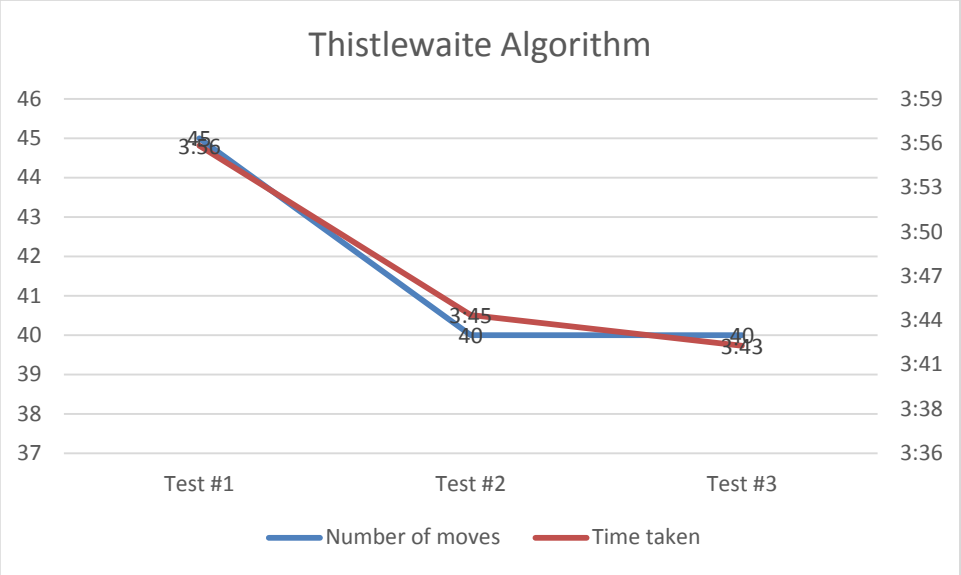
- The Thistlewaite algorithm was the least effective method of solving the Rubik's cube requiring an average of 42 moves and 3 mins and 48 seconds
- The Kociemba algorithm proved to be the 2nd most effective algorithm, requiring an average of 28 moves and 2 mins and 32 seconds
- The Korf algorithm proved to be the most effective algorithm, requiring an average of 20 moves and 2 mins and 5 seconds

As seen in the data, the more efficient the algorithm was, the less amount of moves it took to solve the same jumbled Rubik's cube, therefore yielding shorter completion times.

Analysis

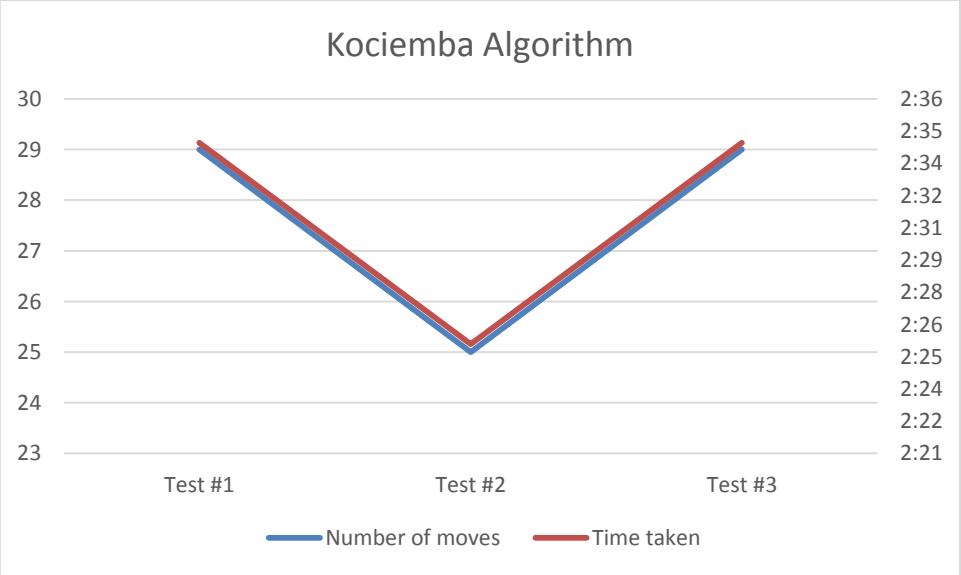
Thistlewaite Algorithm

Test number	Number of moves	Time taken
Test #1	45	3:56
Test #2	40	3:45
Test #3	40	3:43



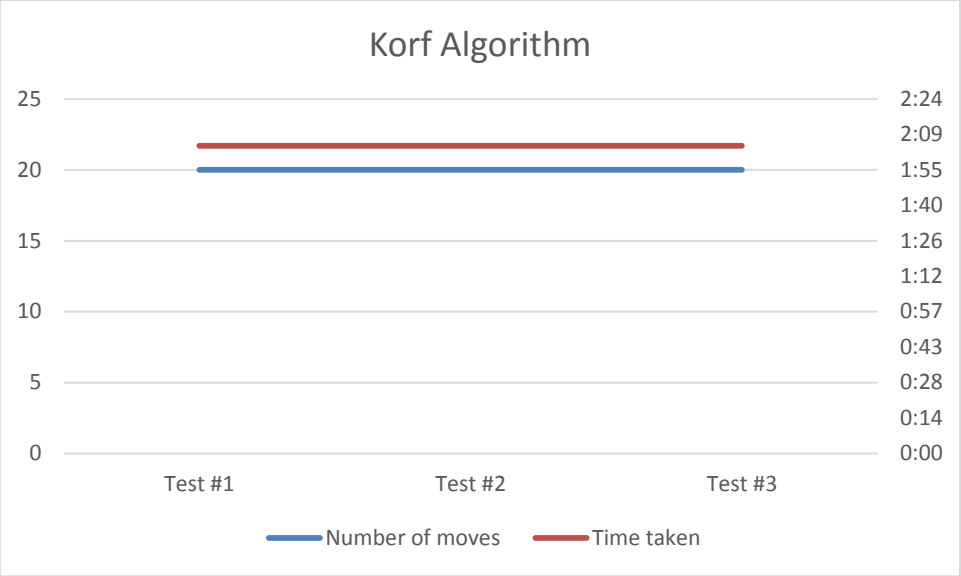
Kociemba Algorithm

Test number	Number of moves	Time taken
Test #1	29	2:35
Test #2	25	2:26
Test #3	29	2:35



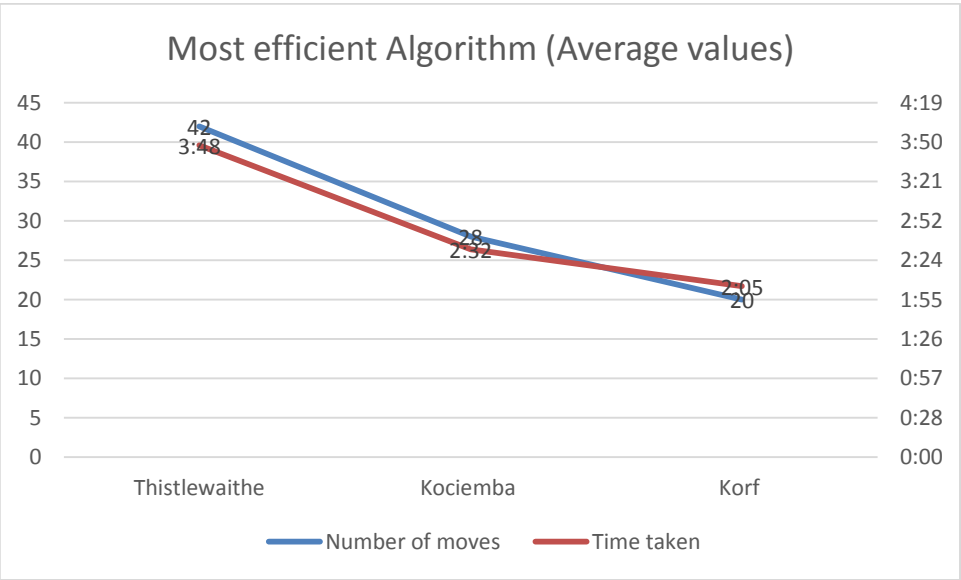
Korf Algorithm

Test number	Number of moves	Time taken
Test #1	20	2:05
Test #2	20	2:05
Test #3	20	2:05



Summary (all 3 algorithms)

Algorithm	Average Number of moves	Average Time taken
Thistlewaite	42	3:48
Kociemba	28	2.32
Korf	20	2:05



Discussion

After conducting the experiment, there are quite evident trends which are present in the experiment results. Across all three algorithms, they each displayed and produced similar, consistent results respectively. As seen in the graphs, the Thistlewaite algorithm was the least effective method followed by Kociemba algorithm, leaving the Korf algorithm as the most efficient. These results were expected based on information gathered online.

As seen in the results, the 3 algorithms performed as expected, due to a variety of reasons:

The Thistlewaite algorithm was the least effective, solving the Rubik's cube requiring an average of 42 moves and 3 mins and 48 seconds. This can be attributed to the fact Thistlethwaite's method differs from layer algorithms and corners first algorithms in that it does not place pieces in their correct positions one by one. Instead it works on all the pieces at the same time, restricting them to fewer and fewer possibilities until there is only one possible position left for each piece and the cube is solved. Thistlewaites algorithm lies in the "upper bounds" meaning it takes more moves to solve. This is due to the fact that the sub problems are only split into 4 subgroups which restricts the amount of moves the equation can be simplified too.

The Kociemba algorithm proved to be the 2nd most effective algorithm, requiring an average of 28 moves and 2 mins and 32 seconds. Herbert Kociemba algorithm managed to combine several ideas into a very effective new algorithm which will give good sub-optimal solutions very quickly. It may well find an optimal solution to a position fairly soon, but it may take a long time for it to actually prove the solution is optimal by trying out all shorter sequences. The first idea was based on Thistlethwaite's work. Kociemba uses only two phases however; this therefore explains the improvement in the amount of moves required and time taken however still not being the most effective

The Korf algorithm proved to be the most effective algorithm, requiring an average of 20 moves and 2 mins and 5 seconds. However it needs to be taken into consideration that the computer connected to the robot, was running for a week prior to the experiment calculating the simplified Korf algorithm, which would normally take 35 CPU years to compute. The expected result can be attributed to the fact that the Korf algorithm is based on a multi-phase coding, which means the equations are divided into numerous sub problems which are solved by the computers CPU. Furthermore, when it is split into tables, normally in the other methods there are a number of ways to reach the solution, however, the Korf algorithm limit the search depths of later phases therefore further optimising the number of moves required, instead of creating multiple solutions. This can be evident in the results, with all three tests of the Korf algorithm yielding the exact same number of moves (20 moves) and time (2mins 5secs) as opposed to other algorithms that did not remain consistent between the tests.

The experiment encompassed the entire experimental concept and met all of the requirements of the scientific research method set. All controlled variable remained constant throughout the experiment. The experiment achieved the aim which was set directly with a fair test. This is re-affirmed by similar results found online showing the same relative trends between the efficiency of each of the algorithms. This proves that other researchers would be able to perform exactly the same experiment, under the same conditions and generate the same results, reinforcing the findings to ensure that the wider scientific community will accepts the hypothesis.

The experiment was repeated 3 times for each algorithm and then averaged out for the final graph the data was extremely consistent with the data showing no major outliers in the results. Although this still quite reliable, This could have been further improved by testing the experiment a larger number of times to re-enforce the reliability, perhaps repeating the experiment 10 times per algorithm, but this was not an option due to time and battery limitations. With each battery requiring 15 hours of charging time and 1 set in the NXT requiring 6, to complete 30 sets of tests would require 450 hours in charging time of batteries alone and require a large amount of money spent on buying rechargeable batteries, this was out of the budget and could not be achieved. Nonetheless the experiment was meticulously conducted to ensure the controlled variables remained the same throughout all the tests. Variables such as using the design of the robot, battery life, programming, jumbled position of the cube and when to stop/start the timer, remained consistent and controlled throughout all of the experiment tests. Furthermore, by making using a robot to solve the cube, instead of merely noting down the moves and solving the cube by hand, it eliminates any room for human error on inconsistencies. This includes, reducing inconsistencies with the time taken to solve, incorrect moves and a range of other human errors, consequently this drastically improves the reliability as the robot performs predictable moves at programmed speeds that remain consistent. If this experiment was to be redone without the limitations of the current scenario, more batteries could be bought and used to increase the test size. In doing so, it would serve to increase the validity of the overall experiment.

The overall level of accuracy in all of the tests was high, this could be attributed to the fact that all measurements recorded were made accurately, such as documentation of moves taken and time taken. Starting the stopwatch and finishing it at the exact right time was planned carefully to ensure a fair test. Once again, the fact that the human interference with the experiment was minimised, greatly improved the accuracy of the test, preventing human errors or inconsistencies. The algorithms used in the experiment were checked thoroughly to ensure it remained as true to the root code as possible when transferring it to the NXT to make sure that the tests gave an accurate representation of the efficiency of the actual algorithm. On the other hand, there are several other strategies implemented in the testing that could further improve the accuracy of the experiment. If given more time, an automatic stopwatch could be programed to appear on the LCD display, not only could this make testing much more convenient, but also reduce the possibility for human error drastically. Additionally, if cost wasn't a problem, the new Lego Minstorms (EV3), would have been able to help with the accuracy of the test by doing all the computing of the algorithms on its own CPU, without connection to a PC. This could mean that the coding would not need to be altered at all to be uploaded to the robot.

In hindsight, there are a range of different limitations that were faced when conducting the experiment that hindered the test results, these either led to compromises in the method, or changes to the experiment to facilitate the limitations. The tests conducted had 3 main limitations which were: cost, time and equipment/software available:

Cost/equipment

In the conducted tests, a Mindstorms NXT set was used to create the robot, however the NXT is a previous generation model which has many flaws as opposed to the new EV3 Mindstorms kit. The EV3 kit has numerous additional features such as upgraded processing power, increased sensitivity sensors, high powered servo motors and many others. At a $500 price, though this was clearly not an option as it was out of the budget. Therefore, it was a much more viable option to use the NXT set regardless of

trade-offs of additional features. Additionally, the high cost of rechargeable batteries was a factor that had to be considered as by buying a fresh set of batteries for each test would cost excessive amounts of money. Therefore, by purchasing 2 sets of rechargeable batteries I was able to alleviate the problem with the trade-off being decreased battery life and a 15 hour charge waiting time.

Time

Due to the high complexity of the experiment regarding, designing, building, coding and testing, time was an issue throughout with only 4 weeks to complete the experiment. This includes the long process of finding the root algorithms online and transferring them to the NXT software, this meant that I had less time to construct the robot, because the robot couldn't be tested without it. Additionally, with only 5 days to complete the experiment, it meant that running 10 tests of each algorithm was not an option due to the long charging time of the batteries.

The tests performed can be applied into everyday life and be used in real life situations to a certain degree. The IDA* Heuristic code is a very common code that is one of the best general-purpose graph search algorithms when there's a way to estimate the distance to the goal. IDA* is extremely beneficial when the problem is memory constrained because IDA* does not remember any node except the ones on the current path it has an extremely small memory profile, as opposed to A*, which keeps a large queue of unexplored nodes that can quickly fill up memory. This is especially helpful in society as this code can help simplify equations with little usage of memory; it is often used in search engines and a range of phone apps.

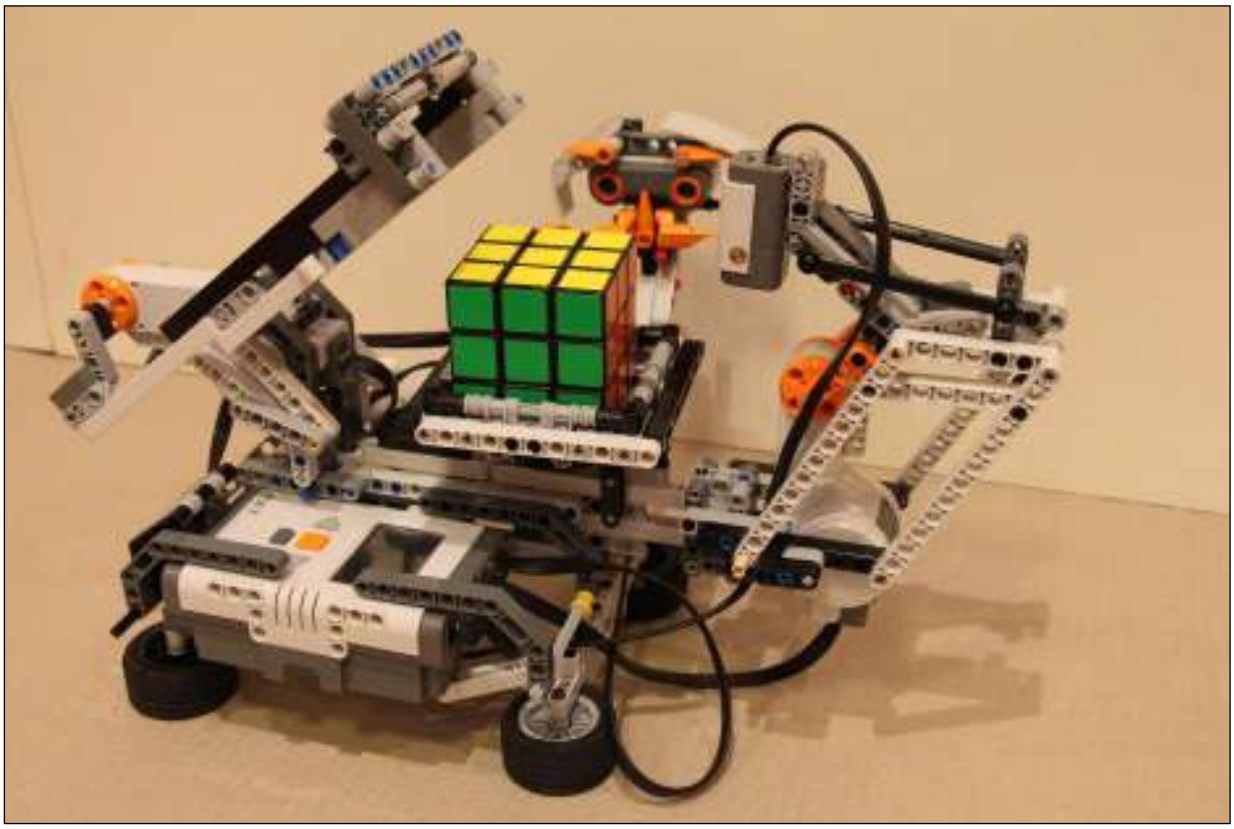
Things to consider: Testing other algorithms to solving the Rubik's cube to see how the compare (e.g. Friedrich algorithm, etc.)? How fast the robot can solve the cube

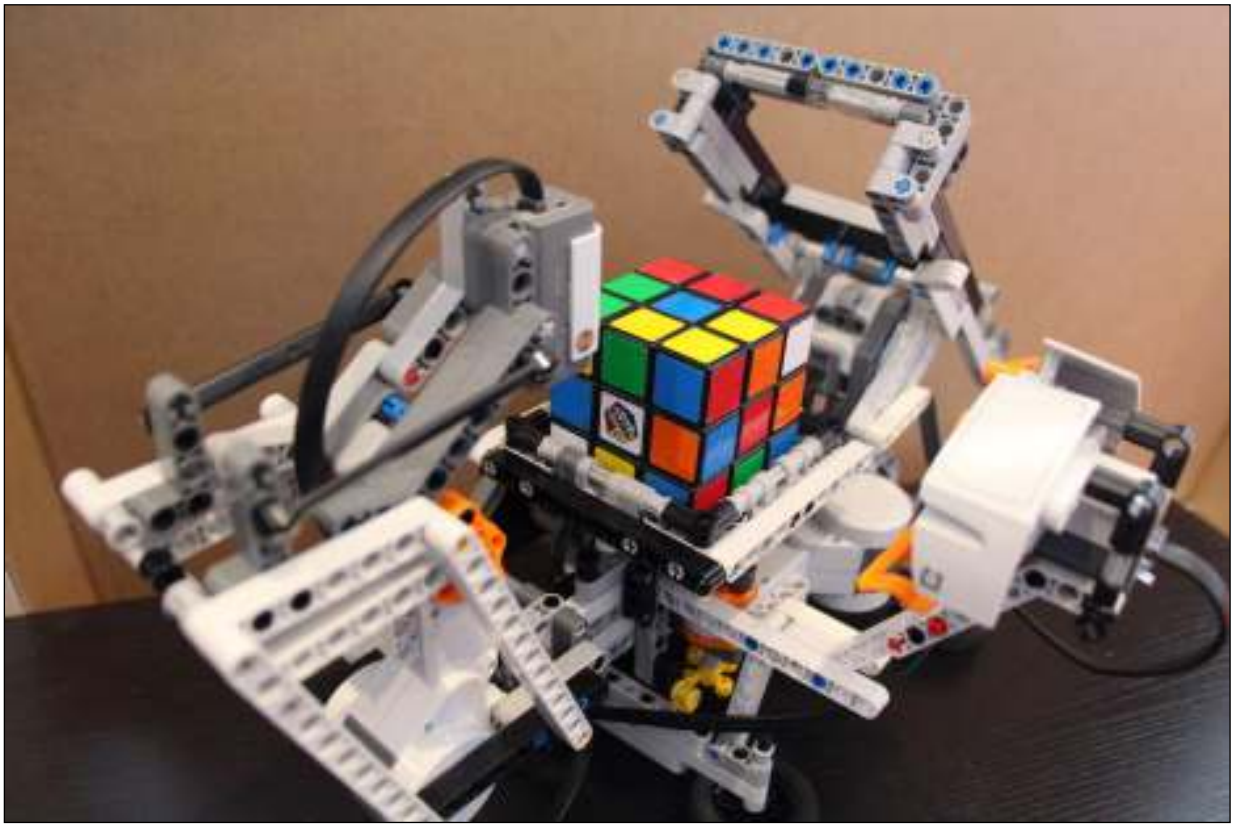
Conclusion

In conclusion, the trends in the data support the hypothesis that Korf algorithm will take the least amount of time and moves to solve a Rubik's cube.

- The Thistlewaite algorithm was the least effective method of solving the Rubik's cube requiring an average of 42 moves and 3 mins and 48 seconds
- The Kociemba algorithm proved to be the 2nd most effective algorithm, requiring an average of 28 moves and 2 mins and 32 seconds
- The Korf algorithm proved to be the most effective algorithm, requiring an average of 20 moves and 2 mins and 5 seconds

This conclusion is based on the data received from a consistent, reliable and accurate experiment. The results were expected as most of sources which were used in the research pointed out the same information.





Acknowledgements

- Oriana Miano (Providing information and assistance on various aspects of writing a scientific report, additionally, providing an experimental report scaffold)
- David Gilday (Mindcuber.com outline of robot design)
- Karen Marcellienus (Proofreading and ensuring coherency of the report)
- Jessica Marcellienus (Photographer of the robot)

Bibliography

- "Algorithm List." - How to Solve a Rubik's Cube. http://www.personal.psu.edu/mjr5125/blogs/how_to_solve_a_rubiks_cube/algorithm-list.html (accessed July 19, 2014).
- "Bob Burton's Cubewhiz.com." Bob Burton's Cubewhiz.com. http://www.cubewhiz.com/notation.php (accessed July 16, 2014).
- "Cube Explorer 5.01." brandeis. http://www.cs.brandeis.edu/~storer/JimPuzzles/RUBIK/Rubik3x3x3/READING/KociembaPage.pdf (accessed July 23, 2014).
- "CubeTwister." CubeTwister. http://www.randelshofer.ch/cubetwister/ (accessed January 1, 2014).
- "Download the Software." LEGO.com Downloads. http://www.lego.com/en-us/mindstorms/downloads/nxt (accessed July 17, 2014).
- "Easiest to code algorithm for Rubik's cube?." java. http://stackoverflow.com/questions/1354949/easiest-to-code-algorithm-for-rubiks-cube (accessed July 17, 2014).
- "How to Solve a Rubik's Cube (Easy Move Notation)." wikiHow. http://www.wikihow.com/Solve-a-Rubik's-Cube-(Easy-Move-Notation) (accessed July 18, 2014).
- "How to solve the Rubik's Cube." How to solve the Rubik's Cube. http://ruwix.com/the-rubiks-cube/how-to-solve-the-rubiks-cube-beginners-method/ (accessed July 18, 2014).
- "IDA*." Heuristicswiki -. http://heuristicswiki.wikispaces.com/IDA* (accessed July 22, 2014).
- "MindCub3r for LEGO MINDSTORMS EV3." MindCuber for EV3 . http://www.mindcuber.com/ (accessed July 21, 2014).

- "Online Rubik's Cube Solver Program." Online Rubiks Cube Solver. http://ruwix.com/online-rubiks-cube-solver-program/ (accessed July 16, 2014).
- Wikimedia Foundation. "Optimal solutions for Rubik's Cube." Wikipedia. http://en.wikipedia.org/wiki/Optimal_solutions_for_Rubik's_Cube (accessed July 19, 2014).
- "Pseudocode Examples." Pseudocode Examples. http://www.unf.edu/~broggio/cop2221/2221pseu.htm (accessed July 25, 2014).
- "Rubik's Cube Notation." WikiCube. http://rubiks.wikia.com/wiki/Notation (accessed July 16, 2014).
- "Rubik's Cube Software." / Programs / Scripts. http://software.rubikscube.info/ (accessed July 27, 2014).
- "Rubik's Cube Solution Methods." / Techniques (Waterman/Ortega/Minh Thai). http://www.rubikscube.info/ (accessed August 2, 2014).
- "RubikCube Compatibility After Effects." AE scripts + Plugins . http://aescripts.com/rubikcube/ (accessed August 1, 2014).
- "Rubiks cube 3x3 solution." Rubiks_cube_3x3_solution. http://rubiks.com/solving-guide/pdf/Rubiks_cube_3x3_solution-en.pdf (accessed July 18, 2014).
- "Tilted Twister." Tilted Twister. http://tiltedtwister.com/ (accessed July 20, 2014).
- "Welcome »." Program to Solve Your Rubik's Cube. http://www.famvangestel.nl/ (accessed August 1, 2014).
- "algorithms to solve rubik's cube." algorithms to solve rubik's cube. http://www.cs.swarthmore.edu/~knerr/helps/rcube.html (accessed July 18, 2014).
- Conde Nast Digital. "iPhone App Solves Rubik's Cube in 20 Moves or Better | Business | WIRED." Wired.com. http://www.wired.com/2009/01/iphone-app-solv/ (accessed July 29, 2014).