C O V E N T R Y
U N I V E R S I T Y

Faculty of Engineering, Environment and Computing

An Investigation of Pathfinding for Navigating Autonomous Vehicles

Author: Euan Campbell (6384171)

Supervisor: Dr Yanguo Jing

Academic Year: 2017/18

Course: Computer Science BSc

Module: 300-303COM Individual Project

Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Science

Declaration of Originality

I Declare that This project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Statement of Copyright

Statement of Ethical Engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below.

Signed:  **Date:** 22/04/2018

First Name	Euan
Last Name	Campbell
Student ID Number	6384171
Ethics Application Number	P67726
1st Supervisor Name	Yanguo Jing
2nd Supervisor Name	Victor Sanchez-Anguix

Contents

Chapter One – Introduction

Abstract

With software always growing in its capabilities and sophistication, we are seeing programs tackle evermore complex problems, replacing many roles that humans have been performing for decades. One such task is driving, where progression in areas including machine learning and Artificial Intelligence have seen vehicles taking to the roads under the control of only a computer. With pathfinding regarded as a crucial asset in Automatically Guided Vehicles (AVG), it is important that the mechanisms driving the vehicles are considered safe and effective for improved predictability in driving situations. Recent road incidents involving self-driving vehicles have stressed the significance of having safe and reliable computer systems where public trust needs to be gained through high performing pathfinding algorithms and vigorous tests carried out on them.

This project carried out a review on current pathfinding algorithms from literature. A new pathfinding method was designed and implemented, using self-generated weights and a wall-hugging approach. The new pathfinding algorithm has gone through a number of tests to evaluate how well it performs compared against existing algorithms. With random maps generated for test variety, simulations were run over large map sizes ranging from 10x10 to 100x100 to understand how the different approaches handled different situations.

The test results show that the new pathfinding algorithm is faster than existing methods such as A* to find a solution. A slightly longer path length is produced however.

Future works include continuing the development of the new algorithm to deal with scalability of the driving environment. 3D simulations with integrated sensors can be used to test the algorithm in a simulated autonomous vehicle driving environment.

Acknowledgements

I would like to show gratitude to Dr Yanguo Jing, Associate Head of School at Coventry University, for the helpful assistance and appropriate guidance throughout his supervision of this project. The efforts to go above and beyond deserve acclamation, including his advice across all areas, both theoretical and practical, as well as the help sharing the idea with industry professionals showing his enthusiasm. The positive project outcome has been directly impacted as a result of his support.

Project Introduction

Algorithms play an important role in almost all computer systems, with a goal of solving a single or series of problems. Their ability when implemented through a computer allows results to be generated much quicker than when done manually, so much so that we have become reliant on them for many aspects of our technological needs.

Pathfinding is an illustration of when such algorithms can be utilised, travelling from a start location to the finish in the best appropriate manner. With methods of pathfinding having remained similar for some time, it felt appropriate to see if any new alternatives could serve as replacements.

With the ability to be applied in areas including vehicle navigation, pathfinding has been used increasingly in the past decade as the rise of autonomous vehicles on a consumer level has become more common. With the rising developments, particularly relating to artificial intelligence and the compact sensors in these vehicles, we have seen the underlying systems become more complex, however the algorithms used for immediate and long-distance navigation remain somewhat similar.

Project Background

The idea for the new algorithm proposed in this project originates from a university project relating to the implementation of search and sort algorithms. There was unfortunately limited time, which meant the initial concept remained idle for almost 2 years, before being developed properly throughout this project.

List of Abbreviations

BFS	-	*Breadth First Search*
DFS	-	*Depth First Search*
NBPA	-	*Node-Based Pathfinding Algorithm*
AGV	-	*Automatically Guided Vehicle*
ROS	-	*Robot Operating System*
CA	-	*Cellular Automata*
MVP	-	*Minimum Viable Product*
CSV	-	*Comma Separated Values*
OS	-	*Operating System*
AI	-	*Artificial Intelligence*
MUPFP	-	*Military Unit Path Finding Problem*

Project Objectives

The project aim is to implement a new pathfinding method, with the potential to be used within autonomous vehicles.

Main Objectives:

1. Implement an assortment of existing pathfinding algorithms for comparison.
2. Design and implement a new pathfinding algorithm.
3. Compare the performance of existing pathfinding methods and the newly proposed algorithm.
4. Investigate the similarities and differences between the pathfinding algorithms.
5. Explore the impact the new algorithm could make to autonomous vehicles.

Chapter Two – Literature Review

The term "automation" was first coined by the Ford Motor Company in 1947 when the company's vice president of manufacturing started their first Automation Department to supplement the manual work of factories, later growing to perform more advanced operations (Weber, 2003). The word was then popularised by John Diebold in his book titled 'Automation', where it was suggested that computers would become the "principle tool" that would drive forward change for business and industrial practices, of which he was not wrong (Ceruzzi, 2003). With automation first concerning the manufacturing of motor cars by Ford, it is only fitting that 70 years later we are seeing automation integrate itself into the vehicles themselves through companies such as Jaguar-Land Rover and Tesla, however now replacing the work of the customer instead of the factory worker.

When John Diebold implied that computers would become a tool for change, describing it as "the buttons now push themselves", it is possible he did not foresee the ability for devices in later years to learn to control multiple buttons. Recent years have seen the rise of terms such as 'machine learning' and 'artificial intelligence', which have introduced a new technological step, allowing computers to learn based on input data and make actions or decisions accordingly (Kisačanin, 2017). Systems such as these have found their way into driving mechanics, including aiding drivers in brake assist through object detection, learning patterns of a driver's preferred interior temperature, and controlling a cars movement for fully autonomous vehicles.

Building Environments

Creating an environment from which an algorithm can be tested provides some challenges when trying to maintain consistency.

Nathan Sturtevant suggested a key issue with analysing grids in his paper "Benchmarks for Grid-Based Pathfinding" (Sturtevant, 2012) whereby there are inconsistencies in the research community when evaluating work on grids due to different problem sets making it difficult to compare results between various papers. This problem is accentuated with the different performance characteristics which are not always clearly apparent in each test set. Sturtevant proposed a standard set of maps for continuity between papers, such that results can be equally compared without accounting for the potential difference in results stemming from various test sets. Although his idea provides map-type variety, it could be justified that there are not enough maps of each type for a full in-depth analysis on pathfinding algorithms. A map of each size exists in only 10 variations. A better approach would be providing rules for which a variety of maps can be created, allowing a quantity to be dictated for use of the researcher, which would better measure the project being undertaken.

When pathfinding is applied to real-world applications, there comes a need for simulating the effects that such a system could have, not solely in a theoretical sense, but also analysing how it could behave in an authentic environment. With pathfinding requiring an entity to be mimicked in a simulation, robotic software often provide the best features, with an abundance of sensors available that can feed data back to an algorithm. Lucas Nogueira compared two robotic simulators, Gazebo and VREP, known for their common use in robotics research, to see which is objectively better (Nogueira, 2014). Both of them were compared in

relation to ROS (Robotic Operating System) integration, world modelling, model modifications, programmatic control, and program efficiency, determining that VREP maintains a higher level of capabilities for early users of robotics. With Gazebo's focus on ROS, it would be more common in areas where a finer degree of control is required for a simulation. With any pathfinding analysis comes limitations in capabilities, which will commonly become apparent in even basic simulations, such that VREP is capable of handling.

Comparing Algorithms

In order to determine effective performance of an algorithm, a comparison approach is often best, allowing for benchmarked results to be compared to one another. This approach was used in Li Yan's paper titled "Performance Analysis of Pathfinding Algorithms Based on Map Distribution" (Yan Li, 2014), where a series of metrics are used to analyse obstacles distributed across game maps. His approach considers distribution information which algorithms such as A* and Dijkstra do not take into account. Therefore, to find the new approaches capabilities, he ran tests on both the new approach and the existing methods to give an idea for how well it compares.

With comparing a new approach against existing ones, comes the need to find appropriate alternative algorithms that show a fitting comparison. Pathfinding is fortunately a well-versed research area, with many algorithms showing consistent use in pathfinding comparison. This is particularly apparent in Khammapun Khantanapoka's research concerning pathfinding where a series of pathfinding methods were tested and compared to see how they fared in multi-layer games (Khantanapoka, 2009). The algorithms used included: Depth First Search, Iterative Deepening, Breadth First Search, Dijkstra's Algorithm, Best First Search, A-Star Algorithm, and Iterative Deepening A*. A number of graphs were created, forming the various layers that when combined produce a finished map. Although his conclusions are limited, having used only four different maps, each with several layers, his approach in studying the various algorithms and the results produced show consistency where a difference of mostly 10%-20% can be seen in the time taken for completion. Using this number of algorithms allows a sophisticated conclusion to be drawn, with algorithms representing a number of uses, showing clearly what impact a new approach can have. When proposing a new pathfinding solution such as this, using similar algorithms would be sensible, however eliminating the need for multiple layers would produce a consistent result, unless the layers themselves are a fundamental requirement for the algorithm.

In 1956, Edsger W. Dijkstra, known for being an early pioneer in computer science, conceived the algorithm known today as Dijkstra's Algorithm (Dijkstra, 1959) with common uses today in areas such as map navigation and the tracking of biological infections (Hooi, 2014). Its significance was summarised by Mikkel Thorup in 1999 in his article 'Undirected Single Source Shortest Paths with Positive Integer Weights in Linear Time' where he stated, "Since 1959 all theoretical developments for general directed and undirected graphs are based on Dijkstra's algorithm" (Thorup, 1999). With the ability to find a shortest path between nodes in a directed graph, it remained unrivalled in some capacity since, making it apparent as to why it finds continued use in situations that require providing a shortest possible route from one location to another. Dijkstra's paper explaining the logic behind the algorithm remains short, with limitations in computing power at the time making it difficult to test the capabilities of

the method. With the widespread availability of computers now, testing can be much more easily performed, allowing for similar algorithms to be compared side by side.

Newer methods of pathfinding have provided increased speeds, depending on the data structures at hand. In 1968, a mere twelve years after Dijkstra created his algorithm, Nils Nilsson, Bertram Raphael and Peter E. Hart took the logic behind Dijkstra's algorithm and established a faster version when trying to implement pathfinding in Shakey the Robot, designed to navigate around rooms (Perry, 2017). Initially being called A2, the algorithm went on to be called A*. Instead of using graph structures, it had more real-world applications, utilising grid-based data structures that could mimic authentic environments, useful for the navigation of devices including robots at the time, but over decades into autonomous vehicles.

Metrics

It can be easy for results to not fully represent the tests undertaken if incorrect metrics are measured. To fully understand how an algorithm performs, a variety of measurements should be collected. Eka Risky Firmansyah looked into pathfinding in relation to Android-based games, comparing a Theta* algorithm to A* in the pursuance of ascertaining which one is more effective. Three important metrics were extracted from the tests, allowing for the conclusion to be drawn. A running time (milliseconds), path length (optimal route), and the number of nodes searched in order to find the optimal route were logged over different tests, which were then averaged for comparison (Firmansyah, 2016).

It is crucial that testing is performed to an extent that accurately represents the capabilities of each algorithm. Each one will have situations in which it can perform better than another, with minimal testing resulting in situations generated which do not accurately depict how an algorithm should normally behave.

In order for pathfinding to be successfully implemented, the importance of having such algorithms which can effectively navigate environments have to be understood. In the case of autonomous vehicles, there is a large move in gaining public trust. Vehicles developed by various companies use safety as a primary reason for developing their vehicles, up to twice as good as that of a human driver (Tesla, n/a), meaning that any faults in their algorithms abilities to navigate around objects that could be potentially life-threatening have to be worked out. Failure to provide effective pathfinding will perhaps result in reduced road accidents, but with no single entity to blame, increased issues will be created as a result.

Daniel J. Hicks provides an interesting perspective on the reality of trusting vehicles that navigate themselves. Public trust is often gained through constant testing and gradual integration into society, until there comes a natural point of confidence where the general population accept the abilities of the new technology. With autonomous vehicles however, it is possible that a real demonstration of AGV's safety will not be available until after they are in widespread use, creating an ironic situation where the masses will not be willing to buy the vehicles, until a large number have already been bought and proved to work (Hicks, 2018). In the meantime, slowly integrating new ideas including the introduction of semi-autonomous features such as Tesla's automatic breaking functionality, allow consumers to learn what vehicles with this kind of technology are capable of.

Chapter Three – Project Management

Project Scope/Risk Assessment

A specific scope is required to give a general idea of the direction of the project throughout.

The scope for this project was separated into three distinct parts, with increasing levels of difficulty, following on from the previous. Working procedurally, each scope can be worked towards, starting at the first, with the second being started, if time permits, having completed a sizeable portion of the first, and starting the third if the same is true of the second scope. With this approach, there can be three potentially completable and separate goals, however accounts for the probability that delays will occur, in which case an accumulation of delays restricts the ability to tackle the scopes in reverse order. As the development process progresses, it will become clearer as to exactly how much of the three scopes can be completed, with changes made to the project plan accordingly to account for them.

The scopes were defined as follows:
Scope 1: A new pathfinding algorithm with four points of travel and a comparison to existing methods.
Scope 2: Improvements made to the new algorithm allowing for eight points of travel with a comparison against the first scope.
Scope 3: Improvements made to the new algorithm with no limited points of travel, with an implementation in 3D simulation software.

A three-step approach was taken, in a manner that was known to produce a meaningful output, even if delays make a large impact.

The first scope is defined with a statement that is feasible, regardless of interferences that could set objectives back, making it is the minimum-viable-product (MVP). With the new algorithm, this is a basic working model that can be used for pathfinding, be it effective or inefficient, but still works in at least fundamental way, even if it cannot compete against the existing methods. The second is a development of the MVP, making small changes that should yield a slightly better conclusion. With more points of travel comes the capacity to travel to more locations in a single step, which, with the way in which the new algorithm works, will only improve its competence. The third an idealistic wish, with an aim that is unlikely to be completed, using the improvements from the second scope to show the new algorithm could have a physical impact on autonomous vehicles, instead of remaining as just a theoretical solution.

Conducting Research

Although there was a basic understanding of the project idea from the start, it is important that there is a knowledge-base that is suitable for the practical aspect. Although conducting early research can be time consuming, it has the potential to save time over the entire project, with a reduced risk of issues occurring that could set the project back, especially those relating to the research methodology.

Completing Tasks within the Time Frame

Although the project was divided into three distinct scopes, a schedule was still created to give an idea for what tasks needed to be completed each week. Some methods from agile development were utilised, making it much easier to know what steps were required to be completed at any given time.

A Kanban board was created using the tool Trello (Trello, 2011), which is designed to keep track of tasks divided into lists that can be moved when a step in the development process has been completed. In a software development project, this will frequently involve lists of tasks being developed, tested, implemented etc, which were transferred to this project, with although a difference existing in that it will not be put in the hands of consumers, the steps involved do share enough similarities.

This tool allowed an overview of the project in each stage, therefore allowing for time to be better managed when changes had to occur. With each task represented by a visual block, it is easy to determine how much time remains until completion.

A proposal was submitted before starting the practical aspect of the project, see Appendix A, where time had to be accounted for in the form of a Gantt chart, showing how each step progresses over time. This chart was changed over time so as to allocate the correct amount of time required to complete the project.

GitHub

Link: https://github.coventry.ac.uk/campb131/Project-AC

GitHub was used for project development, consisting of software that enabled for the storage of the project, with the ability to rollback to previous versions if there was ever an issue. It allowed me to be sure that if there was ever an issue with the new algorithm or project, I could use GitHub to see where it occurred, looking back through commits and changes to find the source. When dealing with time management, it was also useful to look back through how long certain tasks taken, allowing for better prediction of how long future tasks will take.

Supervisor Meetings

With a minimal existing knowledge surrounding this kind of research, meetings were organised with the project supervisor, Dr Yanguo Jing, who could aid in ensuring the project is undertaken in a proper conduct, in relation to the research community. As well as helping define the project into a feasible scope, within the given time frame, he was able to recommend how a conclusion shown be drawn, based on the data that would be generated during testing.

With regular meetings, bi-weekly initially but then every week as the project progressed into more serious stages, he had a sufficient understanding of where the project was at, at any given time, with improved advice given as a result.

A record of all meetings can be found in Appendix C.

Project Presentation

Feedback is essential, with a single point of view within any project not allowing for a range of different opinions that could positively influence the outcome, allowing for alterative points of view to provide additional suggestions that can challenge the methods that have been used so far.

An informal presentation was given to the project supervisor Dr Yanguo Jing, as well as others who he was supervising, as an opportunity to be given feedback in what had been done so far, and what was planned before the project concluded. The presentation can be found in Appendix B. At that stage, the comparison algorithms had been implemented, however the new pathfinding method was still being developed.

Feedback from the presentation included the following:

1. "Keep in mind that the variable limits for the testing should be small enough to be convenient, however allow for the algorithms to be sufficiently tested. Two stages of testing could be used, one for finding the capabilities of the algorithms, and then testing them properly having made a decision based on the testing done previously."
2. "Look into other areas where pathfinding is used, perhaps the new method could be used elsewhere, other than in autonomous vehicles."
3. "Be careful that your comparison algorithms are representative of their abilities. Better and worse variations of the same algorithm can exist, which when tested extensively can show massive differences in their results."

As a result of the feedback, some minor changes were made to the project.

(1) Knowing that little thought had been given in regard to the boundaries in which the algorithms will be tested, an initial testing plan was implemented. This would involve running the algorithms for an undecided period of time, until it becomes slow enough that minimal data is being generated for the amount of time being put into it. At this point, it was unsure if memory limitations would come into play before slowing down occurred to an extent that would involve stopping the tests. From then, the data can be analysed, with results showing diminishing returns allowing an easy decision to be made as to where testing should be stopped.

(2) The projects main focus up to this point was autonomous vehicles, and although a conclusion to this should still be drawn, it posed an interesting question as to whether other areas could be influenced by a new pathfinding algorithm. With the focus on vehicles however, it was decided that this would be done, only if time permitted, with no sense in pursuing an area when fundamental project objectives still need completing.

(3) Although the possibility of good and bad implementations had been thought of previously, there had been some confusion as to what could be done to avoid issues arousing from this. After a discussion with those offering feedback, we decided that only after comparing results would you be able to know if an algorithm has performed well or not, with the only other option involving finding or writing multiple implementations. Time became a limitation here, with the new algorithm being of prime importance and therefore any issues with other

algorithms had to wait until testing is being conducted, where changes could be made accordingly.

Ethical Issues

When applied to autonomous vehicles, pathfinding becomes a consumer-facing solution, with direct effects impacting how passengers are transported. Within the transportation industry, there are ethical concerns for which have to be accounted, to ensure the safety of those using the infrastructure. With road traffic deaths decreasing significantly over the past decade, from 3,409 in 2000 to only 1,713 in 2013 (Department for Transport, 2014), it is vital that autonomous vehicles only improve our safety on the roads.

An incident involving a self-driving car developed by Uber in 2018 where a 49-year-old was hit and killed in Tempe, Arizona (BBC News (No author), 2018), shows that these cars, at their current stage of development, will not eliminate deaths completely while human error is still a factor, be it in the form of other drivers or pedestrians. The investigation for the incident occurred is ongoing.

The only method to ensure navigation without error is continued development in safe environments, where public trust will only be gained through seeing the technology perform as expected over a long period of time. Transferring a task that has been performed by humans for the last century into one that is automated will take time, with older generations less likely to be as enthusiastic to the change.

Improved navigation can affect other areas than those that are consumer facing, such as military applications. The Military Unit Path Finding Problem (MUPFP) involves finding a path from one location to another, avoiding threats with a digital representation of the terrain (Leenan, 2013). Although it is a trivial problem, it does bring about issues surrounding the removal of human intervention in war situations, with pathfinding allowing for improved driving and flight where no person is required to sit in the vehicles themselves.

Issues have stemmed from drones used in the military, where an unmanned plane is controlled from the ground, with some people claiming that it removes the human element from attacks performed using this technology (Hruska, 2018). It could be expected that this would get only worse as improvements to pathfinding and autonomous vehicles could remove even more human interaction when actions such as this are performed.

Through replacing humans with automated processes, be it on the roads, in manufacturing or in areas not yet developed, the ethical implications of the technology developed should be accounted for, where there is potential for negative impacts to grow.

Chapter Four – Implementation and Justification of Methods Used

Program Architecture

The architecture was an important consideration. It was not known from the start of the project how much could be achieved, given the limited time. It was possible that not all project objectives could be met, therefore the program itself had to be capable of working fully, even if some functionalities were lacking. The best method for avoiding any issues with this is through an interchangeable design, whereby modules are written that can be swapped in and out.

It was also important that the program remained as modular as possible to enable change throughout development. This would allow for elements to be developed completely separately before later coming together for debugging and testing. Any issues with the code could therefore be more easily found and fixed through knowing exactly what each block of code should do in a black-box approach where only the output of the code is important.

For this to be achieved, the program was broken down into sections, which come together for the testing itself. In Figure 1, the system architecture can be seen.

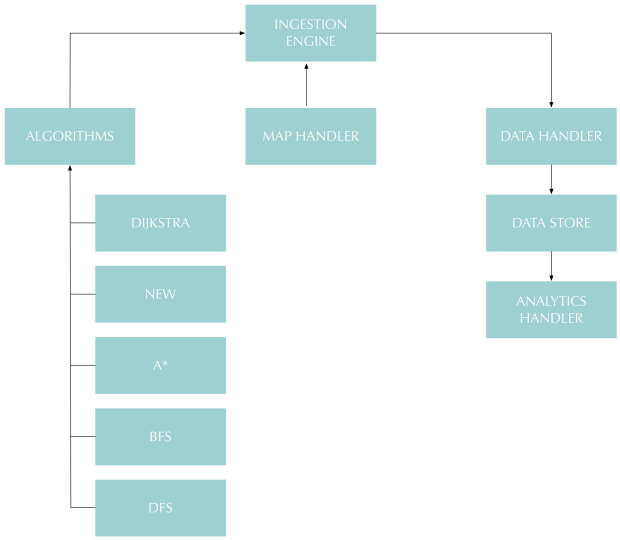


Figure 1 – Program architecture with arrows representing the flow of data

Ingestion Engine

This where the tests are run and exists as the handler for the entire program. It calls the various algorithms and funnels the data from the Map Handler for solving. It exists solely as a place for all the different modules to come together for processing.

Algorithms

Each algorithm is stored as a separate file and called into the ingestion engine when required, normally one after the other for the solving of a specific map.

Map Handler
The handler is responsible for generating and converting maps, also written in a modular fashion so that additional map types could be added and adjusted separately if they were required. The maps themselves are produced one by one meaning that instead of returning all the maps in one go for testing, the handler will instead generate one at a time to allow for flexibility in the initial testing to understand exactly where the program limitations occur.

Data Handler
When data is generated, is needs somewhere to go. It is up to the Data Handler to control exactly what data is wanted and then control the formats and storage so that it can be later processed to be used for analysis, where conclusions can be drawn.

Data Store
Remaining relatively simple, the data store consists of a series of CSV files containing all the relevant data generated by the testing. This data is then pulled out and used for analysis by the Analytics Handler.

Analytics Handler
With the data existing as just numbers in a file, it is imperative that something is actually done with it. A test plan was initially written, allowing this handler to understand what has to happen with the data to draw conclusions.

The New Pathfinding Approach - Methodology
The new approach did not exist as a completed idea before this project was undertaken. Some logical steps in its approach have been implemented with little success, and the main bulk of existing knowledge revolving around its ability to navigate maps with fewer objects, instead being a completed pathfinding methodology that could actually work for all maps.

The proposed new approach for pathfinding, although unique in its ability to find a finish location, utilises ideas from other algorithms in order to work effectively. Dijkstra's algorithm is well known for using graphs with weights between nodes indicating the cost of travel. This new method also uses weights, however instead of being given them, they are self-generated, with the cost instead indicating the chance of having an object nearby.

The methodology for the algorithm can be broken down into two separate steps.

Step One – Directional Importance
The first step encompasses its ability to generate an importance for each of the directions from the current location of the travelling object. This involves analysing the surrounding four locations (north, south, east and west) and finding out, when standing at their locations, how many different positions can be travelled to. From here it is possible to determine which locations will take you into more open space away from walls and which could be dead ends. If a value is equal to four, any location can be travelled to, meaning that there are no surrounding walls, whereas if it is equal to one, then no other location can be travelled to, other than the one where the object is currently standing. After this, the values are given some context. The four values are multiplied by the difference in their axis, meaning that if one axis has a proportionally larger difference than the other, it should be focused on more

and therefore should have a greater chance of being selected for being reduced. If the difference in *x* and *y* are the same, the directional importance bears no real impact on the choice.

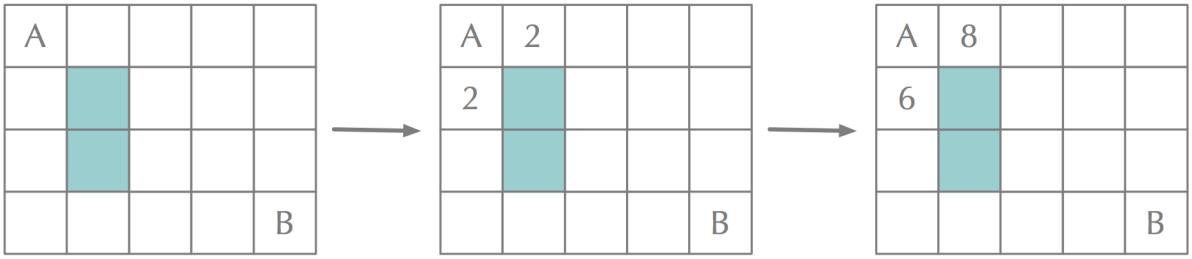


Figure 2 – Methods used by the new algorithm to analyse surrounding locations

$$Difference\ in\ x = 4$$
$$Difference\ in\ y = 3$$

$$Directional\ importance =$$
$$north = 0\ x\ 3 = 0$$
$$east = 2\ x\ 4 = 8$$
$$south = 2\ x\ 3 = 6$$
$$west = 0\ x\ 4 = 0$$

In Figure 2, it shows the two processes explained previously. The first image shows the map as it is given to the algorithm to be solved. The second shows how the algorithm analyses the surrounding locations indicating that both locations have a total of two positions that can be reached from them. The third then generates the importance by accounting for the difference in the axis. From then it can be determined that travelled east is the best position to go to next.

Step Two – Wall Hugging

It became apparent early on when developing step one that the algorithm in its current form would be capable of solving some maps, but not all of them. The approach fell in to the trap of only being able to solve maps where either the x or y can be reduced with each step, whereas some maps will require either or both of the differences to be increased, before being decreased once again. This is common particularly in maze-based maps where there is not a direct path.

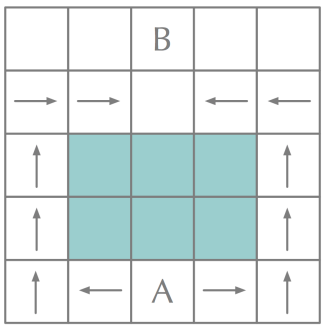


Figure 3 – Two-pointer system for wall hugging utilised by the new approach

Although it took some deliberation, a wall hugging approach was deemed the best method for getting around any walls the algorithm would be stuck behind. This method, often used in maze-solving approaches, requires holding out your right hand onto a wall within the maze, and then following that wall until the exit is found, which works for almost all mazes except for some unlikely and trivial situations that would not exist in the maps generated for testing the new algorithm. This method was implemented through using two pointers, one for left and another for right, each of which taking an opposing side of the wall from the current location and following that round until both the *x* and *y* have been reduced, indicating that the object has been avoided. This is shown in Figure 3. If the two pointers meet, it is likely that no solution can be found.

Step by Step:

1. From the current position, the north, east, south, and west locations are given a value representing the number of travelable points they can reach, which will range from 0 to 4.
2. Each direction is multiplied by the difference in its axis, so west and east are multiplied by the difference in *x*, while north and south are multiplied by the difference in *y* to give them a directional importance.
3. The position with the highest directional importance is chosen to be travelled to.
4. If it is deemed that the algorithm is stuck and cannot get any closer to the final location, two pointers are created which will travel in opposite directions, following the line of the wall it is stuck on until both the *x* and *y* are reduced.
5. These steps are repeated until the finish location is found, or the two pointers meet, indicating that there is no method for getting to the finish location.

Supporting Data Structure

As the new algorithm travels around the map in search of the finish location, it is important that information is kept track of so that it can be used later if required. A data structure was designed to keep track of specific data, including locations that have been travelled to, so that if backtracking occurred, the algorithm would understand not to attempt a path that has already been travelled through and failed. Unfortunately, the algorithm developed away from requiring backtracking, reducing the importance of the data structure, potentially slowing down the running of the algorithm.

Chapter Five – Testing and Data Collection

Testing is perhaps the most important part of a project such as this. The new algorithm, without any kind of testing, exists only as a theoretical idea with no understanding of the potential applications in how it handles various situations. Testing allows for measuring the algorithms success which, when done correctly, enables extensive conclusions to be drawn

Although testing is crucial, it must be performed correctly for an accurate and exact conclusion to be produced. Imperfections in the testing methodology will introduce issues that have the potential to influence the results, making the outcome very different to how it should be.

The modular nature of the system allowed for testing to be easily implemented into the system. The various modules were brought together into one place, where a script could run through all the tests specified.

Comparison Algorithms

It was determined that a comparison approach for results analysis is the best method for determining the new algorithms capabilities, with a series of tests allowing various situations to show where the weaknesses and strengths lie. This approach will also mean that any results generated will be given some context. Performance results for algorithms can vary depending on a variety of factors including the specific implementation of the algorithm, the language that is has been programmed with, and the hardware that it is running on.

All the research into existing pathfinding analysis showed that the best way of understanding the success of an algorithm is to compare it to other, similar algorithms. This gives context to the new or existing approach to know how they perform when given the same problem, showing how one method might be better in one area while another excels in another.

Due to the time limitations of the project, a fairly limited number of algorithms were used for comparison, however they are ones frequently associated with searching maps and have been common algorithms found in existing research into pathfinding.

In an optimistic situation, the new approach would be compared to the algorithms used by the systems that run the autonomous vehicles currently in production. Unfortunately, the organisations developing these vehicles do not release the inner workings of their systems, so common algorithms are instead the best method for comparison.

Dijkstra's Algorithm

This approach has common applications in finding shortest paths within graphs, which are frequently used within long term road navigation. Although consumer maps such as those provided by Google and Apple are likely to include modified versions of Dijkstra's, the fundamental theory that allows for a shortest path to be found will rely on the research conducted by Edsger W. Dijkstra (Dijkstra, 1959).

Complexity: $O(V^2)$

Pseudocode:

```
function Dijkstra(Graph, source):
       dist[source]  := 0
       for each vertex v in Graph:
           if v ≠ source
               dist[v]  := infinity
           add v to Q
      while Q is not empty:
          v := vertex in Q with min dist[v]
          remove v from Q

          for each neighbor u of v:
              alt := dist[v] + length(v, u)
              if alt < dist[u]:
                  dist[u]  := alt

      return dist[]
  end function

}
```

(Abiy, 2016)

Breadth First Search

Traverses maps horizontally, analysing an entire tier of nodes before making its way to the next. In a tree structure, will analyse from top to bottom.

Complexity: $O(V + E)$

Pseudocode:

```
starting vertex x
mark x
list L = x
tree T = x
while L nonempty
    pick vertex v from front of list
    visit v
    for each unmarked neighbour w
        mark w
        add to end of list
        add edge v-w to T
```

(University of California, 2015)

Depth First Search

Traverses maps in a downward motion, using an exhaustive approach with backtracking to engage with nodes at a far reach first. Can be said to work from right to left across a tree representation.

Complexity: $O(V + E)$

Pseudocode:

```
dfs(vertex v)
    visit v
    for each neighbour w of v
        if w is unvisited
               dfs(w)
               add edge v-w to tree T
```

(University of California, 2015)

A Star

Using a modified version of Dijkstra's algorithm, A Star has seen a separation from graph structures and can instead use array maps. Using a heuristic function, it gives a priority to the nodes that should have a better path.

Instead of being used in long distance navigation, it has been frequently found in vehicle simulation. Erik Nordeus' used a hybrid A Star method when simulating an autonomous vehicle (Nordeus, 2016), allowing a simulated vehicle to travel from one location to another.

With use in direct navigation, A Star provides the best information in knowing the extent to which the new pathfinding method could be used for autonomous vehicles.

Complexity: $O(n)$ but can be $O(V + E)$ where a constant heuristic is used.

Map Generation

Existing research into pathfinding analysis showed differences in the approaches that were taken, specifically to the generation of the data structures that have algorithms solve. Some thought it best to have a series of maps that remain the same, while others considered the use of procedurally generated maps instead.

Some initial testing of existing algorithms showed that the results produced could easily vary with small differences in the maps. Differences of up to several tenths of a second were measured in variance from the same map sizes but with slight changes in the start and finish locations, maybe due to greater chances of going down routes that were much slower. As such, it was decided that in order to produce the most accurate results, maps should be generated randomly, allowing for a greater assortment than could be produced manually. It is now possible for results which, when averaged, should come much closer to their actual performance instead of reflecting the small aspects of a map which could greatly impact the output.

Map Types

Extensive research had to be done for map generation, because although it seemed like an easy task at first, issues quickly occurred with regards to how to keep them random, but not so that they become impossible to solve. As a result, different map types were considered for the use of testing.

Cave Maps

Utilising ideas from Dana Larose's "Using A Cellular Automata (CA)Style Rule to Create A Cave System" (Larose, 2002), where she proposed a simple idea using a completely random map at first, with each position having a 50/50 chance of either being a wall or not, before running that output through cellular automata cycles to form the final map. The rule is stated as "I use the 4-5 rule for adjusting squares. This means that for any given square, if it has 3 or fewer adjacent wall squares (counting all 8 cardinal compass points), the square 'starves' and becomes a floor. If it has greater than 5 adjacent wall squares, the square becomes a wall.

Otherwise, leave it as is". Figure 4 shows the two major steps in generating a map, the random generation and the outputting from having passed through the cycles.

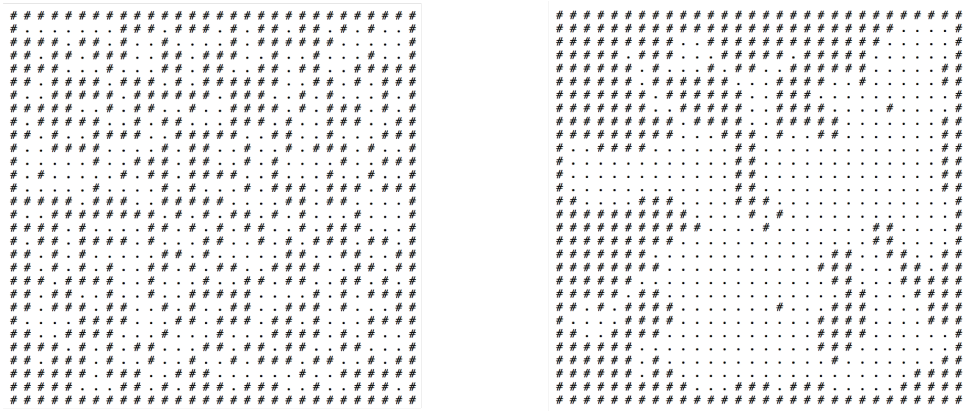


Figure 4 – Random map generation, from the first step (left) to the last, having been passed through automata cycles (right)

It was important that not only are maps randomly generated, but also that the size of maps can be changed to make them increasingly more difficult, testing the algorithms to see how they fare over increased distances. With the CA rules not being limited by the size of the map, a change in the array start size was all that was required for a wide range of map sizes to be produced, pushing the limitations on to the capabilities of the algorithms instead of the handler's ability to produce maps.

The maps themselves are highly relatable to real-life environments. With the purpose of the project surrounding vehicles, it is useful to understand how the new algorithm would handle obstacles of a range of size and shape. Although it is moderately subjective to say, the maps that are generated reflect a similar environment to one that an autonomous vehicle could be put in to. Roads could have been generated, however the pathfinding involved is less algorithmic and instead is focused more on other aspects including keeping vehicles within lines on motorways or similar roads.

For these reasons, cave generation was selected to be used in testing, using the ideas suggested by Dana Larose for creating the maps that the algorithms will try and solve.

Maze Maps

With the ability to only travel in a limited number of directions, mazes exist as a very different concept compared to cave generation. Although there are plentiful methods for generating them, there are some limitations that do not make them ideal for this situation.

They can often be trivially solved, with a common being the right-handed approach, where a wall from the maze entrance is traced until it eventually reaches the finish point. Maps with a trivial solution such as this provide special cases where an algorithms performance compared to others will vary drastically.

Mazes are also not representative of a real-life situation, specifically with autonomous vehicles. In the attempt to find a pathfinding solution that could be used in vehicles that drive themselves, it is important that any simulations reflect similar environments to those

encountered by such vehicles. With roads and open spaces being the common setting, mazes do not represent these well, which would produce results that bare a lack in context to what the new algorithm is trying to solve. It is possible that it could perform much better or worse than the existing methods, but if it will never be used in that context the results become meaningless.

Map Limitations

With the cave generation approach utilising cellular automata cycles, there are limitations. The rules dictating how the map is created accounts for a map with eight degrees of travel, meaning that from a given position, an object can travel to any one of the eight surrounding locations. With the new algorithm and some comparison algorithms only having four degree of travel, maps are generated that are not solvable by the algorithms.

The method by Dana Larose accounts for the fact that it is possible for isolated caves to exist, particularly in larger generated maps, meaning that no pathfinding algorithm would be able to get to the finish location with no path there from the start position. A function is introduced which solves this by making the connections from the separated caves into the middle of the map, however this is done by using eight degrees of travel, so algorithms with only four are not competent enough to travel through these routes.

Figure 5 shows the limitations of using four degrees of travel.

Figure 5 – Representation of degrees of travel, with greater travelable locations on the right, compared to the left.

Formats

In an ideal methodology, each of the algorithms would accept the same map input for easy comparison, in practice however this is much more difficult. Various algorithms rely on different data structures to store the information about a map, meaning that each algorithm had to be fed a different version of the same map, as seen in Figure 6. With the output of each of the generated maps being a two-dimensional array, these had to be converted to the other types so that they could actually find a path.

New algorithm	Two-dimensional array
A Star	Two-dimensional array
Dijkstra	Weighted Graph
BFS	Unweighted graph
DFS	Unweighted graph

Figure 6 – Map formats required by each pathfinding algorithm.

There are two main formats that are used during testing, array-based and node/graph-based. Maps firstly exist as a two-dimensional array, with each value in the structure representing a single position in a grid of uniform fashion, before having to be modified for use by other algorithms.

In Figure 7, a 5x5 grid can be seen, with blue locations defining objects and all other positions being free to travel to. The objective is to get from point *A* to point *B*. With some of the algorithms requiring a graph-based structure instead, the maps have to be converted from arrays into an adjacency matrix, which contains all the connections in a map and is used by Dijkstra's algorithm. Figure 7 shows how the 5x5 grid is converted to a graph, with each position now existing as a separate entity and links in different directions to surrounding nodes.

The graph structure can be seen as more versatile, with navigating arrays posing more challenges due to more checks having to be set in place to ensure that the correct positions can be travelled to.

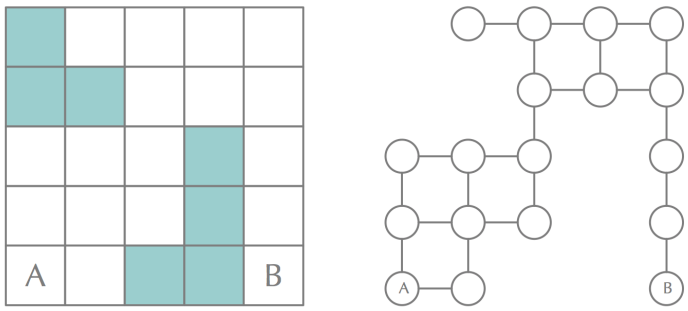


Figure 7 – Visual representation of converting an array map to a graph/node map.

Start and Finish Locations

With pathfinding existing as getting from location *A* to *B*, it has to be decided exactly where those two positions are. Like the generation of the maps themselves, it is important that the start and finish points are also randomly generated.

Methods for approaching this problem range from very simple solutions to those with a greater difficulty which will try and make the best use of any given map. With the map generation not set in place at this point, a simple approach was taken so that it could be changed later if needed.

Using the random library built into Python, a number is generated which corresponds to a location on a map. This location is checked to make sure that it is not a wall, a process which is repeated until both the start and finish location have been assigned.

An issue with this however is that inconsistencies in solutions to maps can occur as a result. A large map could, as a result of the random assignment, have the two points only a few moves away, while a smaller map might make better use of its size. This could mean that large maps are generated, but not fully utilised.

The best approach to this often comes with trying to find the furthest two points on a map away from each other, however this introduces a new issue of the start and finish locations always having to be on different axis and will always be set against a wall. Although the random approach may not be the most elegant or efficient solution, it does give a variety of challenges to the algorithms. Despite this, the unpredictable mannerisms of the allocation should not interfere with the results if they are run enough times, only making the map sizes perhaps correlate to a different average distance.

Results Storage

It was decided before initial testing was performed that a storage solution would be necessary, due to the nature of results being generated and needing to analyse them at a later time. Although it would be possible to draw conclusions from scripts implemented into the tests themselves, it would involve re-running the entire test suite if any issues occurred with the analysis.

Comma Separated Value (CSV) files are a format uses plain text files with an ability to store data in a tabular format, ideal for keeping records. With each test carrying its own separate data, such a format allows for a new row for each test performed, making analysis much easier to perform with easy data retrieval.

Two separate files were used for storing data generated by each test, one for holding any data relating to the map of each test, and another for holding the performance results.

File 1 – maps.csv	***File 2 – results.csv***
Test Number	Test Number
Map Size	New Time
Iteration	New Path
Start (Y/X)	New Path Length
Finish (Y/X)	Dijkstra Time
Ideal Path	Dijkstra Path
Completion Time	Dijkstra Path Length
Array Map	BFS Time
	BFS Path
	BFS Path Length
	DFS Time

Chapter Six – Experimenting and Testing

Testing as a whole was separated into two distinct parts in order to make the final results more conclusive. With independent initial and actual testing, the parameters used to determine the boundaries in the final tests could be better defined, with an understanding of the behaviour allowing for the final project scope to be within reasonable limits.

Initial Testing

Consisting of little formality, the first tests performed were a series of experiments allowing for a better understanding of how each aspect performs. At first, random map size ranges were chosen with different iterations, becoming more refined over time, later developing into the parameters used for the final testing.

Patterns and issues were also able to be found earlier on, without them appearing so much in the final results, influencing the conclusion. Finding one algorithm that did not work and knowing the impacts that bad performance could have on the results allowed for changes to be made later on in the real tests.

Actual Testing

Testing was broken down into three stages:

Testing 1.0 gives a general view of all the algorithms where the testing range sits with an upper bound of a map size of 40, small compared to 2.0. Issues were found and understood, which allowed for changes to be made going forward.

Testing 2.0 exists as more refined testing, with reduced algorithms and a larger test range, with tests run up to a map size of 100, where a better and more extensive conclusion can be made.

Testing 3.0 was used as a special case for understanding the testing environment, recognising limitation restraints in the testing methodology.

Sample Data

At 8.5mb and 138,400 unique tests run, the raw data can become difficult to handle with its vast size. Below is a sample of the post-testing data, before it was analysed.

Test Number	newTime	newPathLength	dijkstraTime	dijkstraPathLength	dfsTime	dfsPathLength	astarTime	astarPathLength
1	0.00028205	9	0.00018597	12	2.69E-05	11	0.00043082	9
2	0.00012803	5	0.00013709	6	2.79E-05	18	0.00017285	5
3	8.70E-05	4	6.99E-05	5	8.11E-06	5	0.00014496	4
4	0.00015283	7	0.00018287	10	3.10E-05	17	0.00031591	7
5	0.00010204	5	0.00046587	12	5.70E-05	34	0.00014496	5
6	7.49E-05	4	0.00040984	5	3.60E-05	29	0.00010014	4
7	7.68E-05	3	0.00046301	4	3.29E-05	24	0.00011492	3
8	9.08E-05	4	0.00036287	11	3.41E-05	19	0.00013685	4
9	7.30E-05	4	0.00013614	7	1.69E-05	17	0.0003531	4
…	…	…	…	…	…	…	…	…

Figure 8 - algorithm_results.csv, where results data was stored

Test Number	Map Size	Iteration	Ideal Path	Completion Time
1	10	1	9	0.000561
2	10	2	5	0.0003221
3	10	3	7	0.00044584
4	10	4	3	0.000283
5	10	5	4	0.0002861
6	10	6	4	0.000278
7	10	7	3	0.00023389
8	10	8	7	0.01059914
9	10	9	3	0.00025702
10	10	10	5	0.00047302
11	10	11	3	0.00038815
12	10	12	3	0.00029397
13	10	13	5	0.00042796
…	…	…	…	…

Figure 9 – maps.csv, where map data was stored

Post-Test Processing

The raw data generated by the tests on its own, does not provide enough information that could be used to draw much of a conclusion. Instead, further processing can be performed that will manipulate the data in such a way that a meaningful inference can be made.

This was performed in two steps, the first being to define what should be looked for, and the second consisting of writing scripts that can perform the tasks required. Six different objectives were written down:

1. Time taken to find a solution
2. Time taken as percentage of all algorithms
3. Path length of the solution by each algorithm
4. Path length as percentage of the ideal path
5. Time taken to find no solution
6. Percentage of incomplete maps

These were then used to construct Python scripts that organise the data into a new CSV file, allowing for the easy creation of graphs which could be later observed to drawn upon an understanding and conclusion.

Test Justifications

Twelve tests were performed in total to develop an understanding of both the algorithms used, including the newly create one and those used for comparison, and the maps generated for the purpose of comparison. Below is a breakdown of each test performed, and its importance in the overall project.

1.0

These tests gave a first look at understanding the initial results and played a large role in making changes that are reflected in the conclusion. The outcome of this first suite of tests allowed a decision to be made that removed Dijkstra's algorithm from further testing due to its poor time performance, and DFS because of its long path lengths.

1.1/2.1

Shows how the path length of each algorithm increases, as the map size increases. This can indicate how well the algorithms can find an effective solution to the proposed problem, with a lower path length showing that a more efficient path has been found.

1.2/2.2

Although a low path length can be found, it must be found in as short a time as possible, with an algorithm producing a high time output indicating that it might produce a good path but could have an inefficient way of finding it.

1.3/2.3

The ideal route is represented by the start location subtracted from the finish location, therefore being the ideal route if no obstacles existed, meaning each algorithm should strive for that amount. The results give a context to how good the output paths from the various algorithms are, with relevance to what the best route could be. An algorithm could have a short path length in comparison to the others, however compared to the best possible route could be very high. Ideally the ideal route would account for obstacles.

1.4/2.4

When running the tests for up to a few hours to generate results, it can be interesting to see how much time is taken by each algorithm in proportion to the others. By showing the percentage of the test time, the resulting graph can show how much time is taken by each

algorithm, and if any in particular use a majority of the processing time. This can be used to remove algorithms if they are shown to be too slow compared to the others, allowing further tests to be run without the slower algorithms for much higher map sizes.

2.0

As a repeat of 1.0 with Dijkstra removed, the upper bound map size could be increased drastically from 40 to 100, allowing for more instances to be generated for more accurate results. A better breakdown of exactly how the new algorithm performs can be seen without the noise from other methods, by comparing it to only its closest alternative.

See sections 1.1-1.4 above for a breakdown of 2.1-2.4

3.0

These tests proposed an alternative point of view through addressing concerns with the tests themselves, by analysing how effective they are and whether they could have interfered at all with the results.

3.1

With the type of map generation used, it was not feasible to generate a solvable solution every time. This test was performed to find out how many unsolvable maps were generated in each test suite, to find out if the tests could have been run quicker or with higher map sizes with an improved method for generating the maps.

3.2

As well as knowing how well an algorithm can find a solution, it is also important to know when a solution cannot be found. This test therefore allows an understanding of exactly how effective the various methods are at knowing when to stop trying to find a solution that is not there.

Testing 1.0

Includes Dijkstra's Algorithm and DFS

Test 1.1.1 Path Length

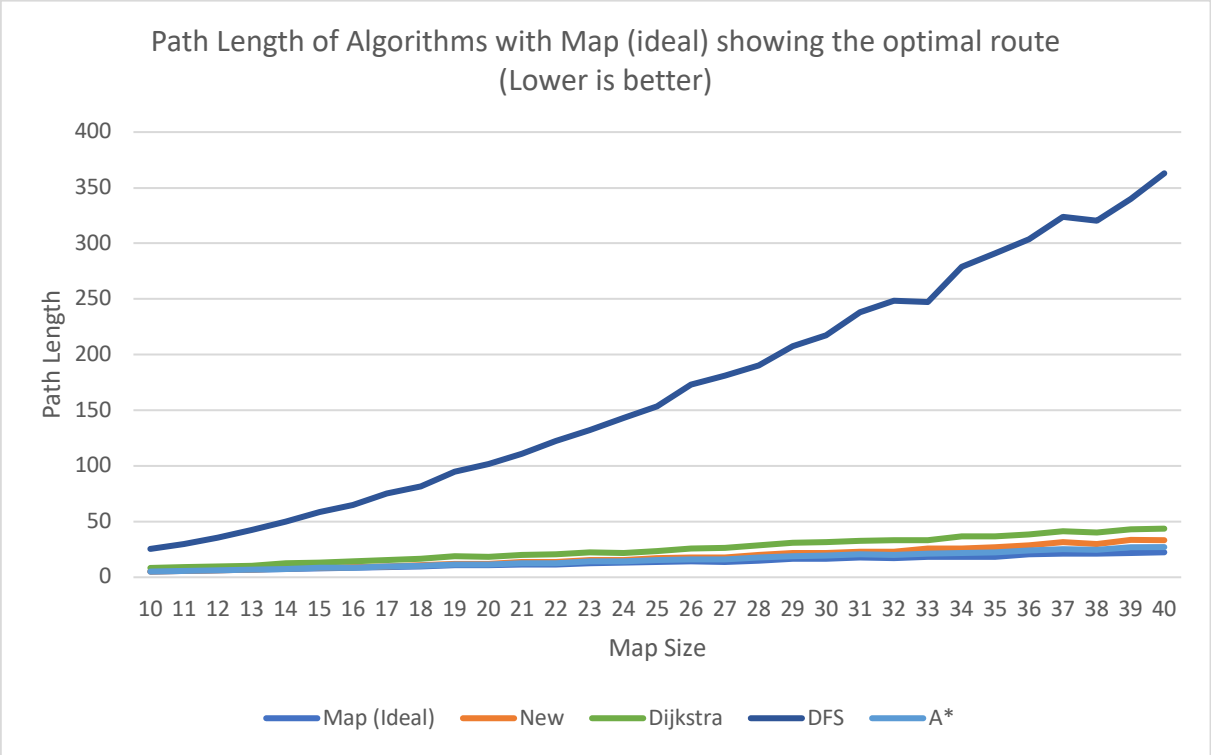


Figure 10 – Comparison of path length against map size

Test 1.1.2 Path Length (without DFS)

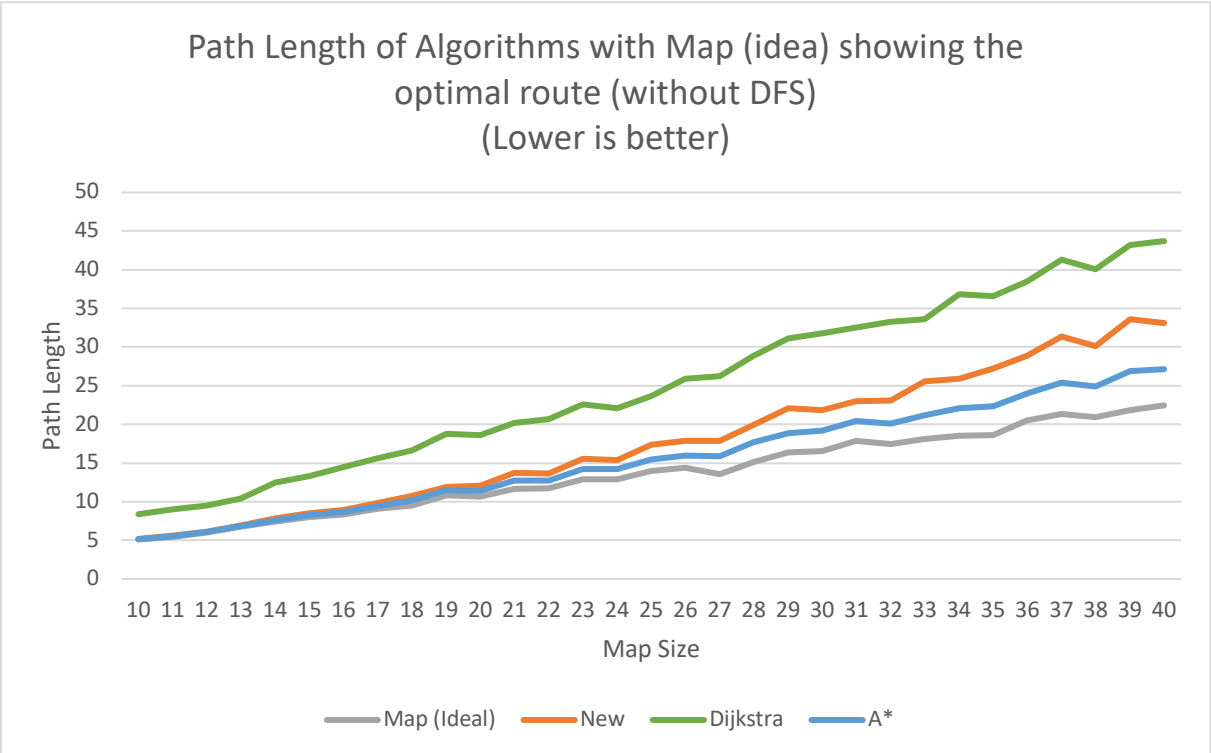


Figure 11 – Comparison of path length against map size

Test 1.2.1 Time taken to find a solution

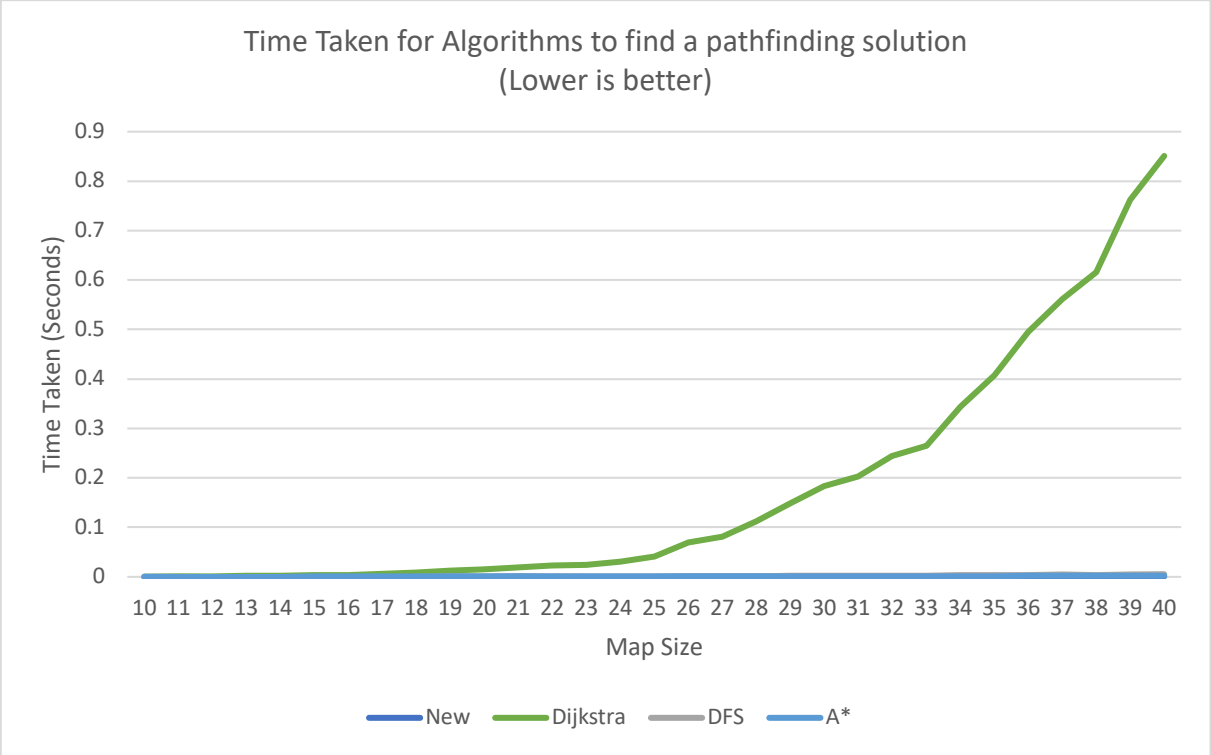


Figure 12 – Comparison of time taken against map size

Test 1.2.2 Time taken to find a solution (without Dijkstra's algorithm)

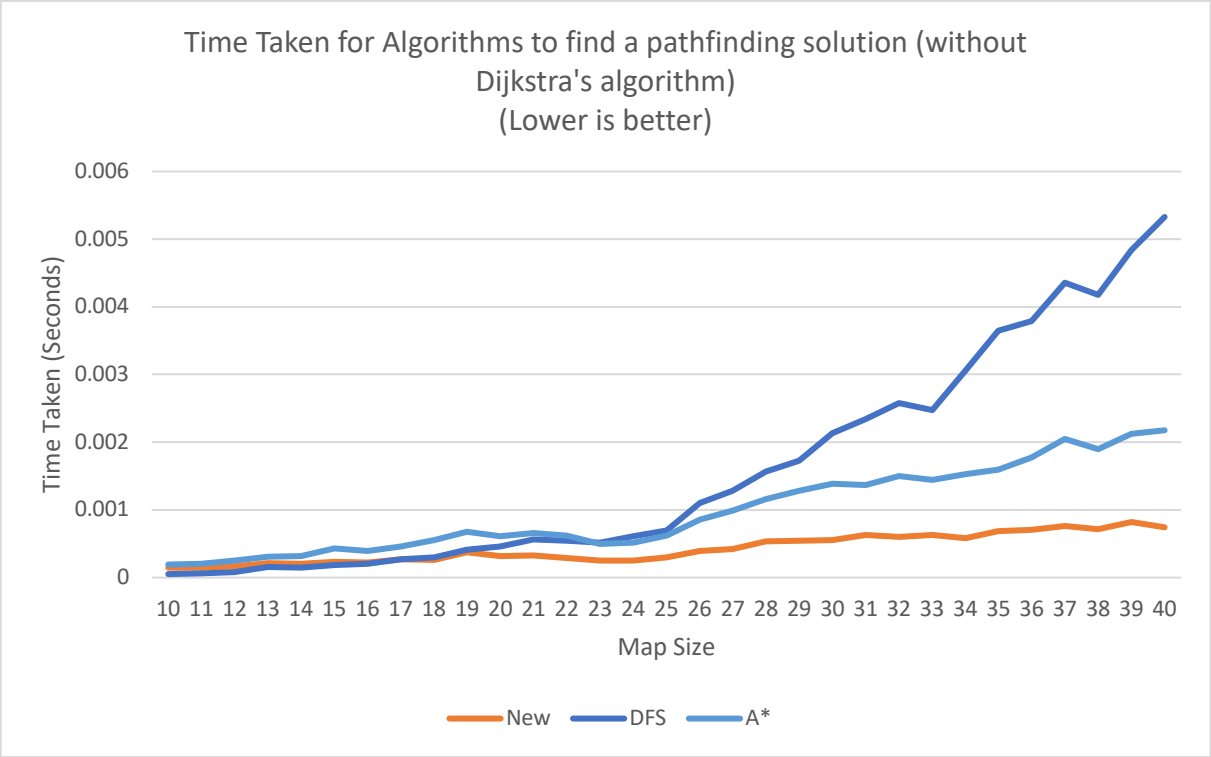


Figure 13 – Comparison of time taken against map size

Test 1.3 Percentage of ideal route

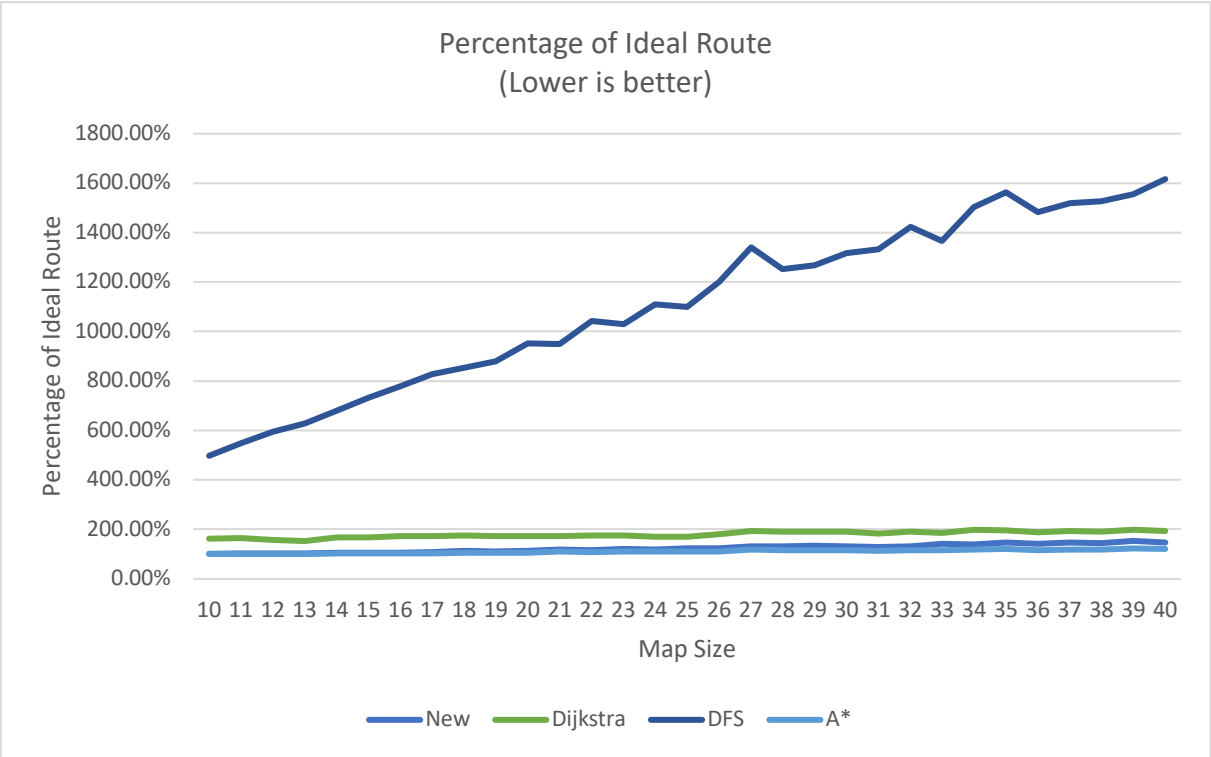


Figure 14 – Percentage of the ideal route compared to the growing map size

Test 1.4 Percentage of time taken

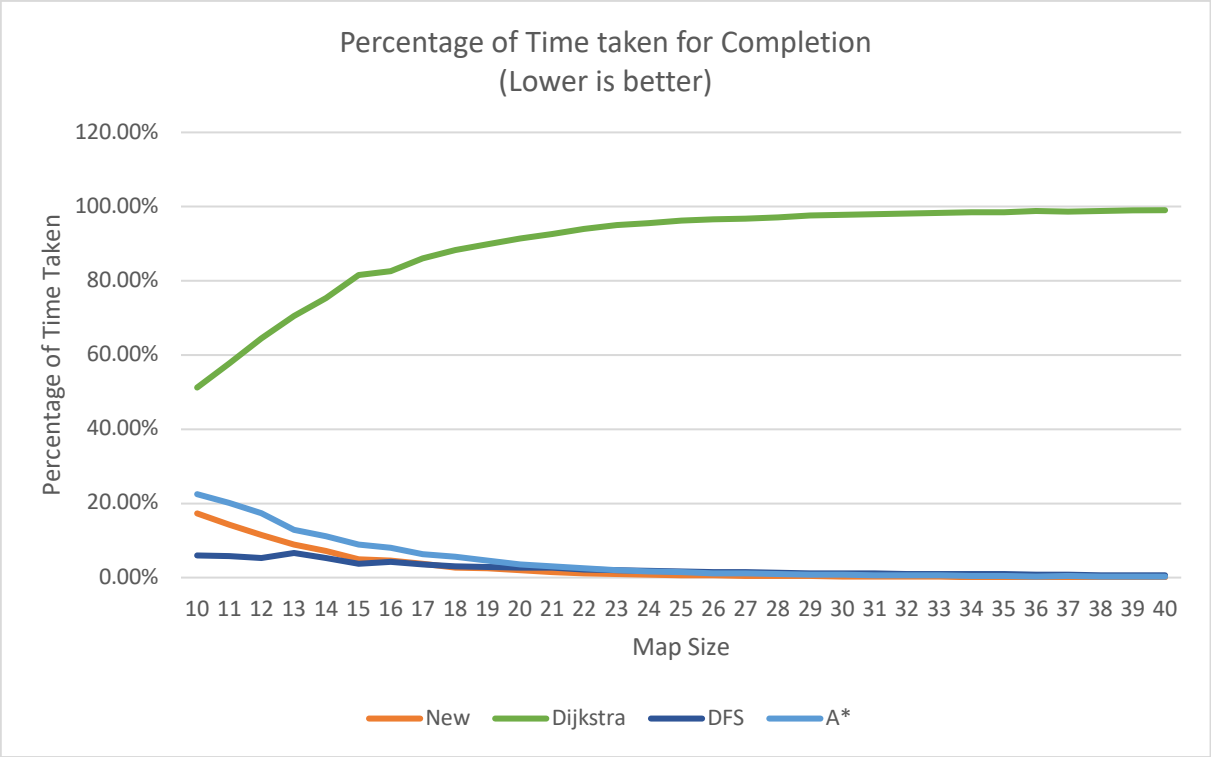


Figure 15 – Percentage of the testing time compared to the map size

Testing 2.0

Excludes Dijkstra's Algorithm and DFS

Test 2.1 Path length

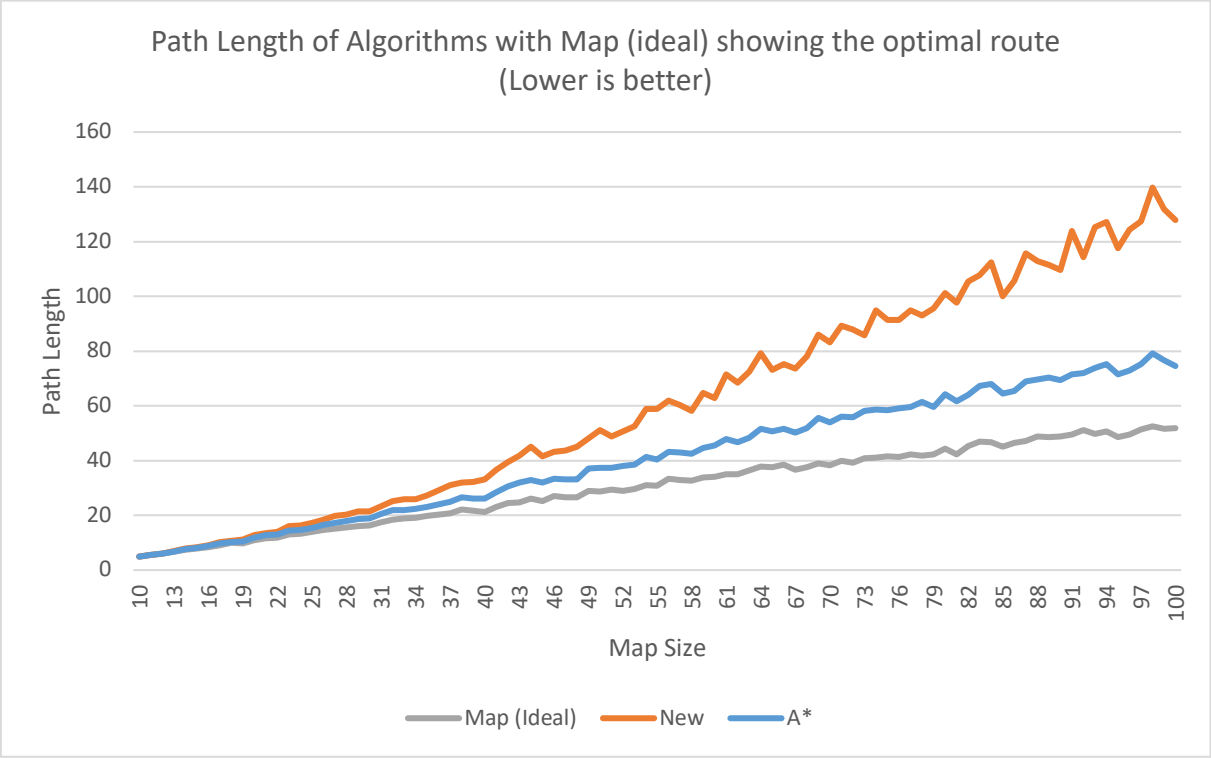


Figure 16 – Comparison of path length to the map size

Test 2.2 Time taken to find a solution

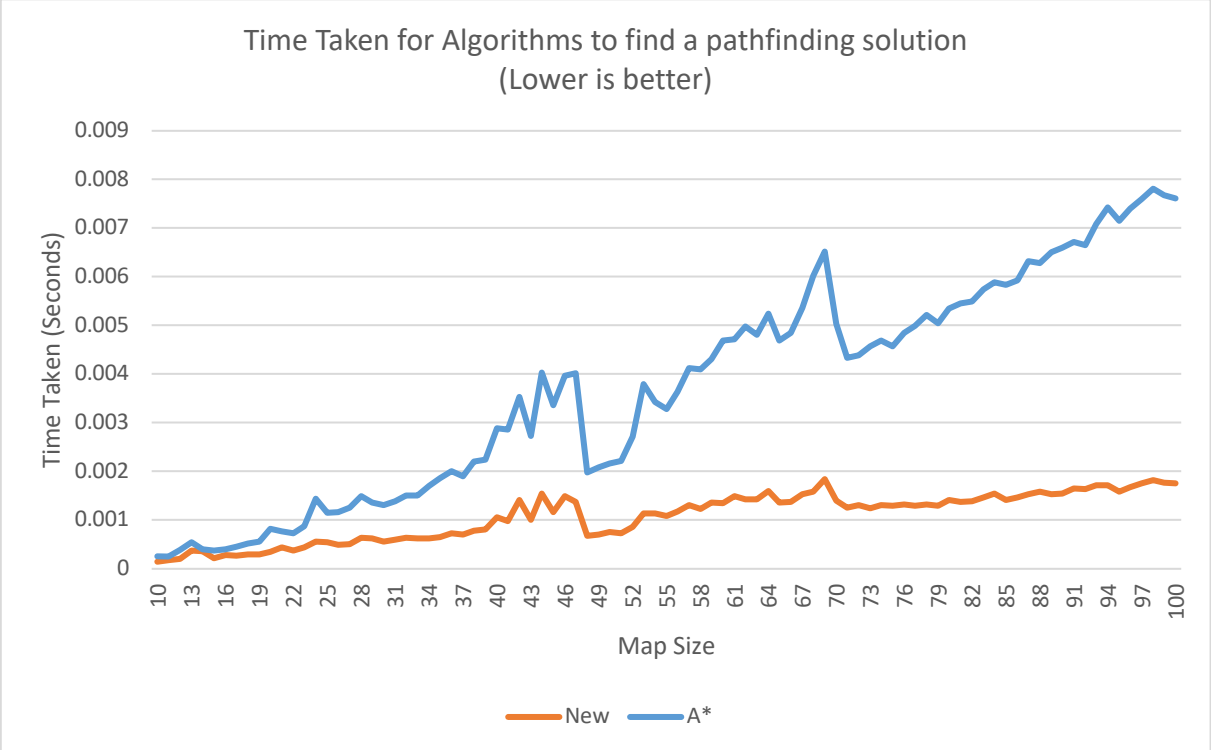


Figure 17 – Comparison of the time taken against the map size

Test 2.3 Percentage of ideal route

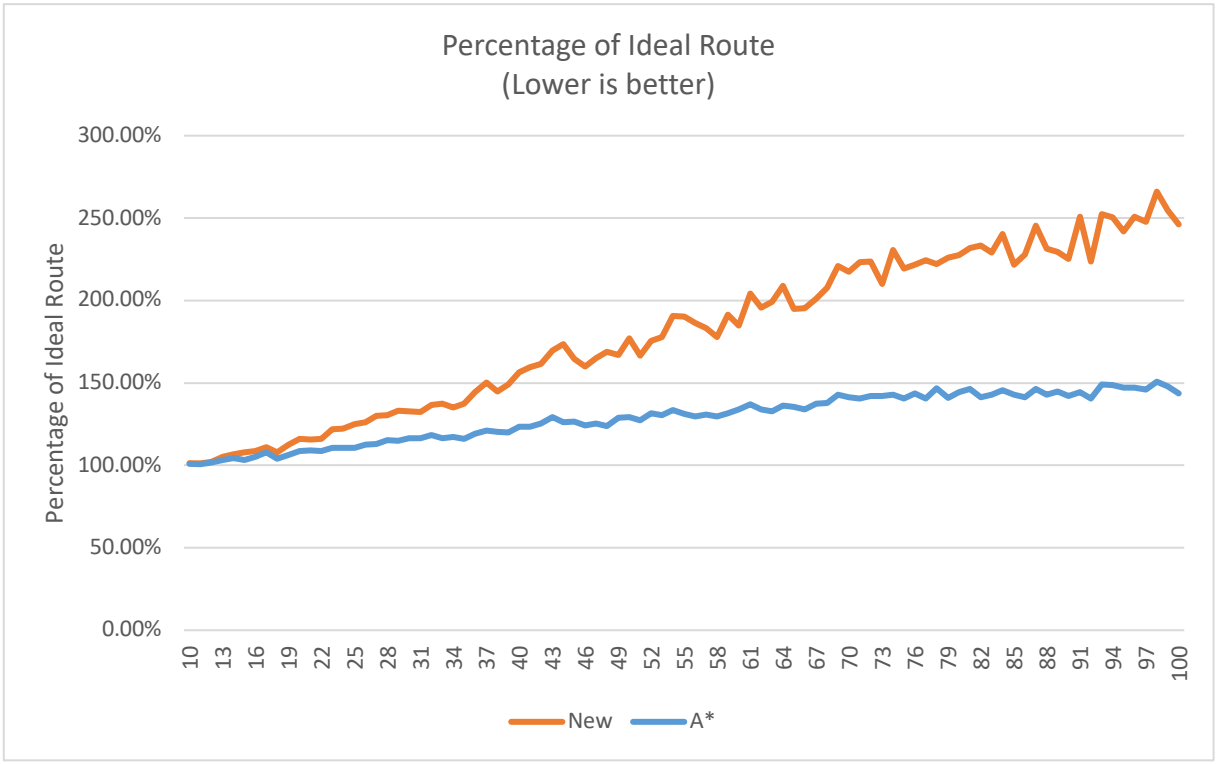


Figure 18 - Percentage of the ideal route compared to the growing map size

Test 2.4 Percentage of time taken

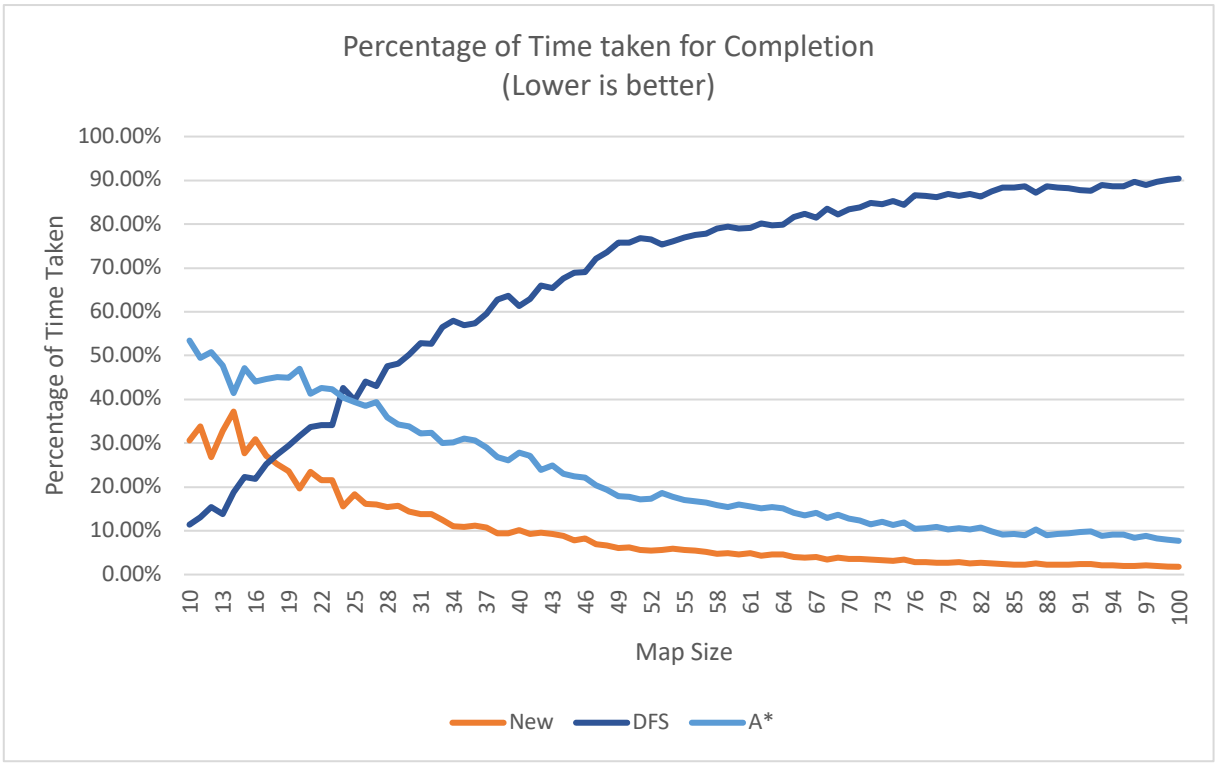


Figure 19 - Percentage of the testing time compared to the map size

Testing 3.0

Unsolvable Maps

Test 3.1 Number of unsolvable maps generated

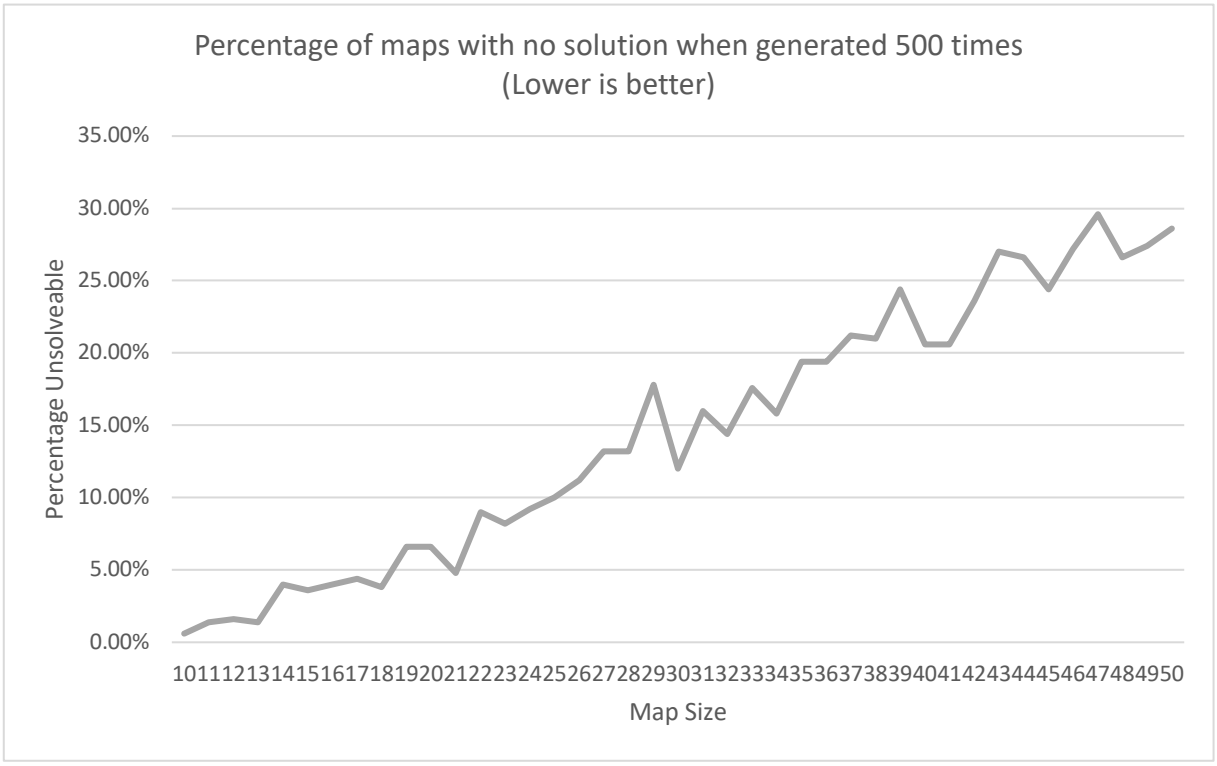


Figure 20 – Finding the occurrences of useless maps in the map generation

Test 3.2 Time taken to know a map cannot be solved

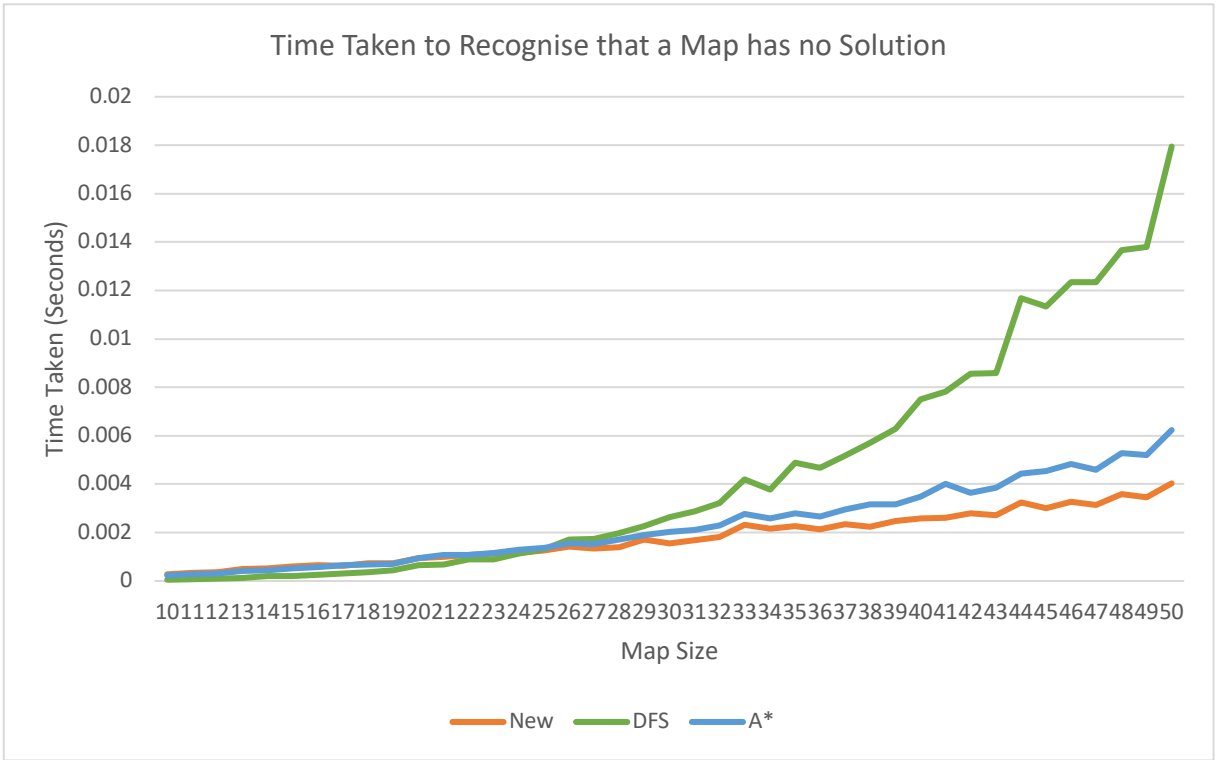


Figure 21 – Tests to find when no map can be found.

Analysing Test 1.1.1 – Path Length, Figure 10

Interpretation

- DFS has largest output at 13.4x the length of next worst algorithm.
- DFS length increases by 256% with each doubling of the map size, starting at 101.9 nodes at map 20, but reaching 362.9 at 40.
- Other algorithms not distinguishable.

Meaning

The graph offers little information regarding any other algorithm, other than DFS, which resulted in tests being re-run without it (see testing 2.0). It is known more for graph traversal instead of pathfinding, which is clearly apparent here.

Take-away

DFS is not an effective means of pathfinding, outputting a path length much greater than any other algorithm tested. Adjusting the analysis is required to know how the remaining algorithms perform.

Analysing Test 1.1.2 – Path Length (without DFS), Figure 11

Interpretation

- Dijkstra has highest path length, reaching 94% above the ideal path at a map size of 40.
- A* maintains the shortest at only 20% above the ideal at same map size.
- New algorithm longer than A* at 47% above.
- Each algorithm appears to be increasing linearly.

Meaning

Each algorithm has a consistent rate of growth, with the longest path length algorithm (Dijkstra) having the highest growth rate and the one with the shortest (A*) having a lower rate.

Take-away

With the graph showing that the gaps between each algorithm are growing as the map size increases, it can be determined that Dijkstra will always return the longest path of the three, while A* contains the shortest and the new pathfinding method sitting between them.

Analysing Test 1.2.1 – Time taken to find a solution, Figure 12

Interpretation

- Dijkstra has worst time at 159x longer than the next worst algorithm.
- Other algorithms not distinguishable.

Meaning

Perhaps down to a programming issue or Dijkstra's algorithm brute-force approach, the time required to find a path is much larger than any other algorithm tested. No other algorithms can be understood due to the severity of performance of Dijkstra's.

Take-away
Dijkstra, in this instance, is not working effectively as a means of pathfinding.

Analysing Test 1.2.2 – Time taken to find a solution (without Dijkstra's), Figure 13

Interpretation
- DFS uses most time by end but starts by using less.
- New algorithm uses less time than both DFS and A* with a lower rate of increase.

Meaning
The new pathfinding approach uses least amount of time once map size increases enough, however DFS in the smaller map sizes actually performs better than the other two optimised methods.

Take-away
DFS is a best-case algorithm for very small maps up to a size of 18 before the new algorithm is the fastest.

Analysing Test 1.3 – Percentage of ideal route, Figure 14

Interpretation
- DFS has highest percentage of the ideal route.
- Dijkstra has higher amount than both A* and the new method, starting at over 150% of the ideal route.
- A* and new approach are similar, with A* being slightly better.
- All are increasing linearly.

Meaning
Matching the results in section 1.1.1, DFS has a path length farthest from the ideal path.

Take-away
DFS performs the worst, with A* and the new approach performing similarly and Dijkstra above those.

Analysing Test 1.4 – Percentage of time taken, Figure 15

Interpretation
- Dijkstra taking proportionally more time as the map sizes increase.
- DFS, A* and the new approach all reducing in proportional time to make room for Dijkstra.
- Dijkstra levels of at just over 99% of the time taken, while the others take a maximum of 0.62% at a map size of 40.

Meaning
Dijkstra's algorithm is always taking a higher proportion of time than the other algorithms and only grows with each increase in map size. With all the other algorithms combined taking less than 1% of the time taken, Dijkstra is by far the least efficient.

Take-away
Dijkstra takes a considerably higher proportion of time, only getting worse with each increase in map size.

Analysing Test 2.1 – Path length, Figure 16

Interpretation

- Ideal route increasing at 80.21% as the map size doubles.
- Each algorithm should try and get as close to this as possible.
- A* increased at rate of exactly 100% across same period.
- New method fluctuated more, particularly in higher map sizes, having a higher rate of growing inefficiency at 164%.

Meaning

The new method cannot maintain the level of efficiency in finding a shortest path that A* can, where there is more consistency with less fluctuating as it increases.

The graph clearly shows how there is a decreased level of predictability in the new method of pathfinding compared to A*, as well as finding a longer path. As the graph size increases, it can be assumed that the new algorithm's uncertainty will increase linearly and is unlikely to outperform A* in finding the shortest possible path.

Take-away

The new method of pathfinding produces a longer path length than A* and becomes less predictable as the map size is increased.

Analysing Test 2.2 – Time taken to find a solution, Figure 17

Interpretation

- Time taken for A* to find solution increases by 245% each time the map size doubles.
- New pathfinding algorithm increases at only 125%.
- Does not take into account the large drops occurring, as visible on the graph.
- A* always taking more time to find a solution than the new method.

Meaning

The new method of pathfinding may not yield a lower path length; however, it can find a path in a shorter time while also functioning more efficiently as the map size increases.

The large fluctuations than can be seen at map size 48 and 69, although causing A* to drop significantly in time taken, make less of an impact on the new algorithm, keeping it more consistent.

Take-away

The new pathfinding algorithm has both a lower time required to find a pathfinding solution and grows at a slower rate as map sizes increase, showing a better time efficiency than A*.

Analysing Test 2.3 – Percentage of ideal route, Figure 18

Interpretation

- New pathfinding algorithm has a higher percentage of the ideal route than A*, increasing too as a faster rate.
- A* fluctuates less, keeping a straighter line.
- A* increases by 11% with each doubling in size, the new method increases by 39% in the same period.

Meaning

A* provides a much more predictable and simple pathfinding solution, staying closer to the ideal path as the map size increases, compared to the new method which fluctuates despite its linear growth.

Take-away

A* out-performs the new pathfinding algorithm when trying to find an optimal route.

Analysing Test 2.4 – Percentage of time taken, Figure 19

Interpretation

- DFS takes least amount of time initially, but grows to take a much larger, overtaking the new method and A* at map sizes of 18 and 24 respectively.
- DFS's time proportion grows in a concave down shape.
- A* and the new algorithm decrease in a concave up shape.
- Both A* and the new pathfinding method decrease in proportional time taken.
- DFS uses over 90% of the time resources at a map size of 100.
- New method takes proportionally less time than A*

Meaning

A* and the new algorithm are decreased due to the time hungry nature of DFS as the map size grows, where it will travel to increasingly more locations.

As also indicated by section 2.2, A* takes more time than the new approach.

Take-away

DFS is a time-consuming algorithm. A* takes proportionally more time compared to the new method,

Analysing Test 3.1 – Number of unsolvable maps generated, Figure 20

Interpretation

- No solution maps increase in quantity linearly as map size increases.
- Increased by 28.6%, a rate of 0.5% per map size, from map sizes 10 to 50.

Meaning

As the map size grows the number of maps that do not have a solution continues to grow making the time efficiency of testing much worse over time.

Take-away

With such a large percentage of these maps providing an impractical and time-consuming hurdle, tests should be adjusted for a more efficient test strategy, allowing for more tests and of a greater size to be run in the same given time frame if each one generated comes with a solution.

Analysing Test 3.2 – Time taken to know a map cannot be solved, Figure 21

Interpretation

- DFS takes up to 4.4x the amount of time taken to understand a map cannot be solved.
- DFS provides worst time, with the new method and A* having much lower times at a map size of 50.
- New method better than A*.
- New method increases by 1355% from map 10 (0.0003 seconds) to 50 (0.0040 seconds).

Meaning

The time taken to find a solution represents a tiny fraction of time, however increases quickly over a small period. More testing would be interesting to see if each algorithm could increase logarithmically (as Dijkstra seems to start doing) with enough data over larger map sizes. Unfortunately, the data required would take much more processing time, however more unsolvable maps are produced from the amount generated due to the nature of how they are made making it computationally faster over time.

Take-away

DFS shows it is slowest at finding that there is no solution, while the new method out performs A*.

Hardware and Software Specification

Hardware

Programming:

- MacBook Pro 13-inch 2017

Testing Server:

- AMD A8-6600K Processor 3.9GHz
- Gigabyte F2A55M-HD2 Motherboard
- 8GB Corsair/Crucial/Samsung PC3-12800 1600MHz DDR3 Memory (2 x 4GB sticks)
- NVIDIA GeForce GT 730 1024MB Graphics Card

Software

- Programming Operating System: macOS High Sierra
- Testing Operating System: Linux Ubuntu 14.04 LTS
    - Running as a virtual machine using Oracle VM Virtual Box on top of Microsoft Windows 10 Home Edition
    - Accessed through a network using TeamViewer
- Development Environment: Atom
- Programming Language: Python 2.7

Evaluating Results

Testing 1.0

Two major elements stand out from this section, which influenced the two major changes in later tests. DFS is a terrible pathfinding algorithm, the results in 1.1.1 clearly show a path that is not close to the ideal, with results from 1.2.2 showing that it does not have a better time efficiency than either the new method or A*. This is for one very good reason however, DFS is not optimised as a pathfinding algorithm, and instead has its main purpose set for graph traversals and retrieving the right information in the right order, while being able to visit every node in a graph. If a problem required visiting every node in a graph, DFS could be an ideal method, however from point to point it serves little use.

Evaluating the results from 1.2.1 and 1.4, it is clear to see the detrimental effects of Dijkstra's algorithm on the entire tests, with it taking upwards of 90% of the entire testing time. From this is would be possible to say that it has little use in pathfinding, however these results could be as a result of the very reasons why it is such a useful pathfinding algorithm. Instead of just trying to find the optimal path from one location to another, it can, in its analysing of the map, have the capabilities of finding other point-to-point paths having analysed only once, whereas each other algorithm would have to start from scratch. It is possible that the implementation of Dijkstra's algorithm, in this instance, played some role in its inefficient time usage, however the methods used in analysing should of themselves take a larger time proportion, particularly in larger maps as the results showed.

Testing 2.0

With the second set of tests being more refined, it is possible to draw more of a conclusion about the new pathfinding approach.

Unfortunately, it is not possible to say that the new pathfinding method yields a better route, with A* having a slighter better path, expected to be mostly due to the heuristic approach used. Due to some simple logic reasons behind the implementation of the new algorithm, there are occasional situations where paths can be extended unnecessarily, perhaps being a major factor in the difference in the path lengths.

Despite not having a shorter path length, it is possible to suggest that the new pathfinding method can find a path faster than A*, whereby the time taken increases at half the rate that A* does. This means that if the required route in a given situation is not too reliant on the best possible path available but instead the speed at which it can be generated is important, then the new algorithm could be a more effective method, given that similar maps are presented in the problem.

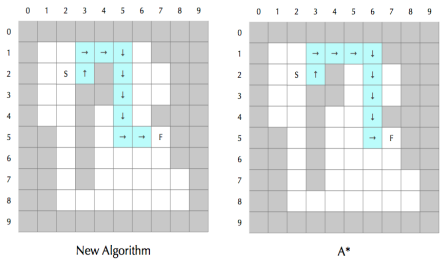


Figure 22 is a comparison of A* and the new approach, showing how they can produce similar results, however they differ due to the new algorithm using a wall hugging approach.

Figure 22 – Comparison of the new algorithm and A in finding a solution.*

Chapter Seven – Reflection and Conclusion

Environment Generation

Inefficiencies

When first suggesting that cave generation could be suitably used for map generation, it was already known that a guaranteed solution would not always be produced each time as a result of eight points of travel being accounted for, instead of four used in these tests. The impact on the initial tests in particular took their toll however as unsolvable maps increased to 29% of the total generated at a map size of 50.

Due to the nature of how the maps are generated and the start/finish locations being allocated, there is an increased chance that no solution will exist as the map size is enlarged. As the tests were run, fewer tests were actually being performed on the algorithms, with graphs generated not accounting for instances where there was no solution, therefore causing fluctuations in the path lengths because of fewer actual tests being counted.

This was later fixed by re-running any tests that did not produce an actual solution, using A*'s path length (the most accurate) as a marker.

Realism

The cave generation was originally seen as a useful way to generate random maps, however with the results having been produced, a different point of view can now be seen.

With the project context existing around autonomous vehicles, it would make sense to assume that a similar environment to those encountered by these vehicles would be the ideal environment for testing. With a lack of methods existing for how to bring about roads into the required format, the scheme used was seen as the best alternative at hand.

Although the reasons given as to why cave generation is a good method of randomly generated maps are still valid, much further testing is required on a variety of maps to really uncover the true value of the newly proposed algorithm. More time would have allowed this, with further scopes enabling even 3D virtual environments where roads could be set up.

Need for removing algorithms

Having a variety of algorithms implemented to allowed for an effective comparison of the new pathfinding method against others, giving it some context. Particular algorithms, however, produced results that were so distant from others that when analysing them became difficult, with three of the four algorithms planned initially removed in the final test suite. The exact reasons for algorithms being removed can be found below.

BFS	Issues with running it in the test suite, whereby it failed to produce any output.
DFS	Considerably longer path lengths than those produced by other algorithms.
Dijkstra	Taking 90%+ of the total testing time, high instance tests could not be run without being bogged down.

Despite being used in existing research, it is possible that the implementation caused issues with the results. Better coded instances of each of them could have yielded better results, particularly Dijkstra's which was expected to perform better.

Comparing against a higher volume of algorithms would have provided a considerably better conclusion, with only one algorithm not contributing much in the context of a comparison approach.

Computing power

As with any project testing the capabilities of an algorithm through testing such as this, a limitation will always exist in that the tests themselves can only be run as fast as the hardware used to conduct them. In this case, a virtual machine was set up with Linux Ubuntu used as the Operating System on a computer with no other open programs, whereby all the tests could be run in a consistent environment, without interferences from other software.

With this is place, the tests were performed until there seemed to be a small yield from the time being put into the tests, whereby each test would take about a second compared to the several hundred performed per second in the early stages. For tests 1.0, a map size of only 40 could be reached, while tests 2.0 were run up to 100 when some algorithms were removed.

Although the results generated provide accuracy, it would be useful to see the comparison on tests run in a more powerful environment for a longer period of time.

File Sizes

A file structure was set in place, so that the outputs of tests would contain the correct data required to derive an objective conclusion. Included in this initially were the actual path outputs of all the algorithms, as well as all the information required to reconstruct each map.

This was done with good intentions so that specific instances could be run again to figure out what was occurring in special instances. The consequence of this was large file sizes, with *maps.csv* growing to *2.2GB*, making it difficult to open in programs such as Microsoft Excel.

The columns containing the ultimately superfluous data were removed, with any important information such as path lengths being added directly instead of calculated later. From this stemmed the ability to run up to map sizes of 100 with smaller file sizes, where before they were stopped in the region of 40-50.

Accuracy of Results

When running tests, a single map was not used for each map size, instead an iteration value was set, determining the number of times that each map size would be tested, before moving on to the next. Through doing this, accuracy can be improved with a greater number of values being entered into the calculated average. With random generation being exactly that, random, it is always possible for unusual occurrence to slip in to testing, with the iterations allowing these to be smoothed out in the final results. For greater improved accuracy, more iterations can be used, which can only increase the veracity.

Conclusion

Although pathfinding is the central topic for this project, it relates to the importance it plays within autonomous vehicles, stating whether pathfinding methods can be improved.

Even having completed testing, it is still difficult to make a conclusion to the problem stated at the beginning of the project due to the word "improvements" ability to be interpreted in many ways, with pathfinding algorithms not having a single metric by which a comparison can be made. With this is mind, a rough conclusion can be drawn, although further research including a greater variety of algorithms, particularly specialised ones, across a wide range of maps would produce a more extensive understanding of the characteristics of the algorithm proposed.

Pathfinding methods can be improved, with the tests conducted showing a reduction in the time taken for a path to be found, however the path length outputted is increased slightly.

In relation to autonomous vehicles, it is not possible at this time to determine whether the new approach could be suitable for taking full control of navigation, with integration of various hardware offering greater complexity than the algorithm is yet ready for. As mentioned previously, taking the algorithm into 3D simulation software and using sensors as inputs for the surrounding locations could deem the algorithm a success in improving autonomous vehicles. At this time however, it solely works better against existing methods, such as A*, which have been tried and tested in real-life situations, in a specific and somewhat limiting testing environment.

In Chapter One, five objectives were defined, stating the purpose of the project and serving as a guide for formulating goals. Below it is explained whether these objectives were completed and to what extent.

- Objective 1 - *Implement an assortment of existing pathfinding algorithms for comparison.*

With four comparison algorithms implemented, it would be easy to say that this was a complete success. The algorithms produced issues however, with only one working to an extent that allowed for a full comparison against the new algorithm, with the remaining three being dropped from a majority of the testing. More research should have been performed to understand better from the start how well each of them would work, with better alternatives chosen as replacements if they were not deemed acceptable.

- Objective 2 – *Design and implement a new pathfinding algorithm*

With tests showing the extent of how the new algorithm performed, it can be said that a new pathfinding approach was implemented, showing performance levels similar to existing methods. Improvements can still be made however, with implementing a greater degree of travel allowing for even faster pathfinding.

- Objective 3 – *Compare the performance of existing pathfinding methods and the newly proposed algorithm*

The results in chapter six show the comparison tests of the new algorithm against the existing methods. Alternative algorithms should have been used, with the majority of ones

implemented producing issues, however the results can still reflect the new approach's capabilities.

- Objective 4 – *Investigate the similarities and differences between the pathfinding algorithm*s
The extensive testing allowed the pathfinding algorithms to be compared successfully, with although some the algorithms not being capable of drawing a definitive comparison, it shows how different methods can produce different results, based on their purpose. With some of the algorithms having poor time or path length performance, it is possible to understand that the results exist because of the defined purpose of the algorithms where in some situations they are best suited.

- Objective 5 – *Explore the impact the new algorithm could make to autonomous vehicles*
With limited time, it was not possible to draw a full conclusion regarding this objective. Further research could show the algorithm has real potential in autonomous vehicles, however in the meantime it can only be said that it can theoretically out-perform existing approaches in simulated tests.

3D simulations, with sensors used as inputs to the algorithm should be capable of definitively concluding whether the algorithm could make an impact to automatically guided vehicles.

Personal Reflection

Although I can deem the project successful, it important that I can reflect on my own capabilities that have been shown throughout the project.

Having been at university for three years studying computer science, programming has played a large part in most projects undertaken. In all of these it has been important to account for how programming fits into the scope of the work required, understanding its influence on the outcome. In many senses, the previous experience allowed for this project to be better managed, knowing the time constraints that come with implementing such a program and the issues as a result. I cannot say that, at times, the programming aspect has been difficult, but I have dealt with these issues appropriately, taking on issues as smaller problems that need solving along the way.

With some previous experience in developing algorithms as part of a university module, I felt confident going into the project, however at times my knowledge was limited, particularly with issues surrounding the implementation of the comparison methods. Comparing against the algorithms did not work as well as originally expected, which more experience could have aided in.

The new algorithm approach has posed some interesting challenges, with a new concept having very little reference material for help, however needless to say it was implemented in a successful manner. The extent to this could be argued however, with the code for this algorithm alone being up to 5x the length of other algorithms used. With more knowledge and programming experience, I have no doubt that improvements to its efficiency could be implemented, even now having finished the project I know several ways in which I would go back and change to cut down on its length.

The project had a real-world application to autonomous vehicles, however, with an emphasis on the pathfinding algorithms and a limited time frame, it was not possible to research this area as much as was planned from the beginning. I would like to look into the subject more so as to later develop the algorithm for such a purpose.

I have always considered myself to be well organised, most of the time. Often with projects it is easy to get out of sync and perform jobs out of order or take short cuts, for the sake of getting it done sooner, or with less effort. With this one however, I can happily say that the management was successful, with strays away from the plan only occurring due to unforeseen delays, which were handled as appropriate. The new pathfinding approach did take longer than expected, which could have been better accounted for, however the project plan was adjusted to keep the project continuing at a rate within the completion time.

I am fortunate enough to know my project supervisor, Dr Yanguo Jing, very well due to a previous university trip abroad. As a result, our relationship has allowed for easy communication and respect has been offered on both sides. Meetings were, as a result, always constructive which allowed for development to continue at a persistent rate.

Chapter Eight – Discussion

Algorithm Improvements

The algorithm in its current state works through utilising a four-point system, whereby an entity can only travel along two different axes in two directions. This may be common in areas such as graph traversal, however in areas such as game development, the ability to travel in numeral directions sees an increased challenge for the algorithm. There are three potential solutions to such a problem:

1. An entity continues to use the four-point method, capable of looking north, east, south, and west, however it can move independently of the map it is travelling on. This would allow objects to still be detected in any direction by any of the four surrounding pointers, with movement translating to being able to rotate instead of simply moving in a solid direction. A speed could then be implemented into this, perhaps travelling at greater speeds in more open areas in a simulated environment…
2. The entity increases its number of surrounding pointers, from four to eight or potentially even more depending on the application. With the increased number of pointers comes more potential locations that can be travelled to. Each step in a cycle would take more time however with a much greater overhead and a lot of time spent processing pointers where most will not impact the movement. It would likely result in more accurate and precise movements, with pointers handled in a much more elegant method than they are currently, with each one being independently programmed.

Looking Past Autonomous Road Vehicles

A project conducted by Ryan Davison-Smith at Coventry University focuses on automating drones for use in a number of applications, namely as a delivery system. With road vehicles being the main focus so far, it is easy to forget that with drones also seeing recent developments with miniaturised technology, they too will require pathfinding as a form of navigation in the coming years (Davison-Pearce, 2018).

With drones, the immediate issue of having to circumnavigate not only the area in two dimensions around it, but also be capable of travelling along a *z* axis as well proposes issues. With some modifications it is possible that drones could use the algorithm, however with fewer objects to avoid it is possible for the new algorithm to slow it down.

Concluding Remarks

The research in this project has investigated the extent to which pathfinding can be improved, with some relevance to autonomous vehicles. Tests were carried out, using scientific practice and a reliable methodology to determine that pathfinding approaches can be improved, with a new algorithm showing it can out-perform a method (A*) currently used in autonomous vehicles when measuring the time taken to find a path. Further research should be done on the new approach to better it and identify its relationship among other methods.

Although it is not proven to work as a method for AGV's, seeing the new algorithm applied to a real-world situation could give the algorithm real promise, with potential applications in a variety of areas where pathfinding is an important consideration.

References

Abiy, T., 2016. *Dijkstra's Shortest Path Algorithm.* [Online]
Available at: https://brilliant.org/wiki/dijkstras-short-path-finder/
Atom, 2017. *Atom IDE.* s.l.:https://atom.io.
BBC News (No author), 2018. *Uber halts self-driving car tests after death.* [Online]
Available at: http://www.bbc.co.uk/news/business-43459156
Ceruzzi, P. E., 2003. *A History of Modern Computing.* s.l.:MIT Press.
Davison-Pearce, R., 2018. *Autonomous navigation of a flying robot (Quadrotor) in the outdoor environment.* Coventry: s.n.
Department for Transport, 2014. *Annual road fatalities.* [Online]
Available at: https://www.gov.uk/government/publications/annual-road-fatalities
Dijkstra, E. W., 1959. A Note on Two Problems in Connexion with Graphs.
Firmansyah, E. R., 2016. Comparative Analysis of A* and Theta* Algorithms in Anroid-Based Pathfinding Games. Volume 6.
Hicks, D. J., 2018. The Safety of Autonomous Vehicles.
Hooi, B., 2014. *Tracing Infectious Diseases using Genetic and Spatial Data.* [Online]
Available at:
https://pdfs.semanticscholar.org/efd6/7368d180b9665fc8e259000c01e084cb8893.pdf
Hruska, J., 2018. *Think One Military Drone is Bad?.* [Online]
Available at: https://www.extremetech.com/extreme/265216-think-one-military-drone-bad-drone-swarms-terrifyingly-difficult-stop
Khantanapoka, K., 2009. Pathfinding of 2D & 3D Game Real-Time Strategy with Depth Direction A*Algorithm for Multi-Layer.
Kisačanin, B., 2017. Deep Learning for Autonomous Vehicles. *2017 IEEE 47th International Symposium.*
Larose, D., 2002. *Using A Cellular Automata Style Rule To Create A Cave System.* [Online]
Available at: http://pixelenvy.ca/wa/ca_cave.html
[Accessed 20 November 2017].
Leenan, L., 2013. A focussed dynamic path finding algorithm with constraints. *Adaptive Science and Technology (ICAST).*
Nogueira, L., 2014. Comparative Analysis Between Gazebo and V-REP Robotic Simulators.
Nordeus, E., 2016. *Self-Driving Car WebGl.* [Online]
Available at: http://www.habrador.com/p/self-driving-car/self-driving-car-webgl/
[Accessed 9 April 2018].
Perry, T. S., 2017. *SRI's Pioneering Mobile Robot Shakey Honored as IEEE Milestone.* [Online]
Available at: https://spectrum.ieee.org/view-from-the-valley/tech-history/space-age/sri-shakey-robot-honored-as-ieee-milestone
Sturtevant, N., 2012. Benchmarks for Grid-Based Pathfinding. 4(2).
Tesla, n/a. *Full Self-Driving Hardware on All Cars.* [Online]
Available at: https://www.tesla.com/en_GB/autopilot
[Accessed 8 April 2018].
Thorup, M., 1999. Undirected Single Source Shortest Paths in Linear Time.
Trello, 2011. s.l.:https://trello.com.
University of California, 2015. *Design and Analysis of Algorithms.* [Online]
Available at: https://www.ics.uci.edu/~eppstein/161/960215.html
[Accessed 2018].

Weber, A., 2003. *Special Section: Automation Pioneers.* [Online]
Available at: https://www.assemblymag.com/articles/83963-special-section-automation-pioneers
[Accessed 7 April 2018].
Yan Li, Z. Z. W. Z., 2014. Performance Analysis of Pathfinding Algorithms Based on Map Distribution. 12(7).

Appendices

Appendix A - Project Proposal

300/303COM Detailed Project Proposal

First Name:	Euan
Last Name:	Campbell
Student Number:	6384171
Supervisor:	Dr Yanguo Jing

SECTION ONE: DEFINING YOUR RESEARCH PROJECT

1.1 Detailed research question

Can pathfinding methods be improved for better navigation of automatically guided vehicles?

1.2 Keywords

Pathfinding; autonomous-vehicles; navigation; graph-theory; simulation;

1.3 Project title

An investigation of pathfinding for navigating autonomous vehicles.

1.4 Client, Audience and Motivation:

Why is the project important?
When developing new ideas, such as vehicles that can drive themselves, various considerations have to be made for such technology to develop over time. The A* algorithm, first described in the 1968 paper "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", is still commonly used today for pathfinding, using an extension of Dijkstra's searching method (Peter Hart, 1968). Despite having existed for this long however, there are still limitations in both of these methods, particularly with regards to the wasting of resources. Dijkstra performs a search 'blind', engaging in a brute force method until at optimal path has been found. In a situation where a path has to be found optimally and resources cannot be wasted, common in self-driving vehicles, waiting for such a solution is not effective enough. To avoid this, alternative methods should be researched for optimal solutions to be found, improving on work from existing pathfinding methods and developing new ideas, using benchmarks to compare them to the alternative techniques.

As a result, exploring these methods can improve the efficiency and effectiveness of pathfinding, possibly being used in real-life applications. This project looks to expand knowledge in this area through the development of a new algorithm that could have the benefit of increased capabilities in pathfinding.

To whom is the project important?
Autonomous vehicles will likely become a norm for future transport, therefore, the central aim of the project is directed towards software developers, specifically those working for organisations developing such technology in this area, where object recognition in relevance to pathfinding in real-life situations has to be accomplished. Most of these companies do not disclose their individual methods of handling pathfinding, however developers involved in the system creation could utilise methods from this project in systems that control getting from point A to B. With the potential to help in areas such as crash avoidance and journey duration, it could be possible to improve the user experience for transportation, pushing the boundary of importance of the project to the users of which the technology will be impacting.

1.5 Primary Research Plan

Step by Step Plan

1) Literature review of pathfinder algorithms and methods
Research into similar methods of analysing the problem, implementing existing ideas where appropriate into the practical element for comparative results.

2) Develop the pathfinding method
Develop the idea of the algorithm. A formal model will be required for a better understanding of how it will work, utilising visual methods to better represent exactly how the algorithm will function.

3) Implement the pathfinding method
A programmed version of the algorithm, in a language best appropriate for the purpose.

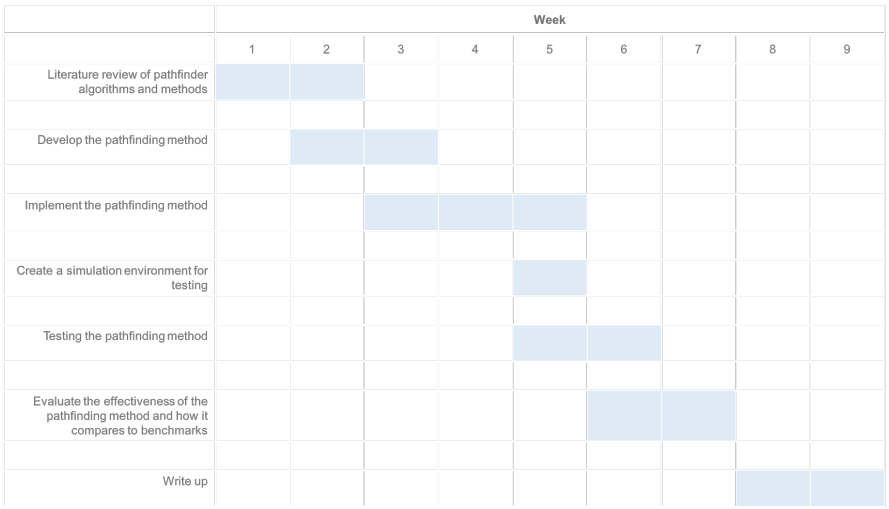
4) Create a simulation environment for testing
With the proof of concept confirmed, a simulation environment will be created, allowing for the algorithm to be tested in a closer to real-life situation.

5) Testing the pathfinding method
Tests will be required both during and after the programming of the algorithm so that it can be confirmed that it works sufficiently at solving a pathfinding problem.

6) Evaluate the effectiveness of the pathfinding method and how it compares to benchmarks
Performance tests will be run on a variety of existing pathfinding methods, with results generated to allow for comparative data, allowing the new algorithm to be benchmarked and judged against them.

7) Write up

Gantt Chart



THIS IS THE END OF SECTION ONE.

SECTION TWO: ABSTRACT AND LITERATURE REVIEW

2.1 Abstract

With the demand for Automatically Guided Vehicles (AGVs) growing each year, guiding them from one location to another has to work efficiently and without fault. The testing of AGVs is ongoing, slowly reaching consumers through manufacturers such as Tesla, using new techniques to consider all possible solutions. Pathfinding algorithms are commonly used for graph traversal but have had their logic implemented in other areas in recent years, such as where the A* algorithm is now used mostly for navigation within maps and games. However, it is a relatively new direction to use pathfinder algorithms in AGVs. This paper focuses on finding a new pathfinding solution that will produce an optimal route, using minimal prior knowledge of obstacles; detecting and avoiding them in real-time. A basic grid-method will be used initially, developing into better simulations with increased recognition points once the algorithm is confirmed as working. With each new step in the algorithm's development, testing will be conducted, allowing for a comparison against the heuristics of existing pathfinding solutions, with feedback consisting of the time taken for completion.

2.2 Initial/Mini Literature Review (500 words – 750 words)

A History of motor cars and Automatically Guided Vehicles

In the summer of 1886, the first motor car took to the streets of Mannheim, Germany, with an engine designed by Carl Benz and a potential that would continue to grow over the next century (Daimler). 130 years later, we have begun to see not only vehicles that are hugely more useful and widely owned than the originals by Daimler, but are also now capable of driving themselves.

At the beginning of 2018, the International Consumer Electronics Show was held in Nevada, United States, where various companies had the opportunity to present their development technology to the media (Warren, 2018). Fisker Automotive, Mercedes-Benz and Ford all presented ideas for how they would lead the new generation of vehicles that would require no driver, instead allowing all those in the vehicle to be passengers. Although we are seeing this new technology working on our roads, with recent tests in Coventry, England by Jaguar Land Rover (BBC, 2017), it is apparent that we are still some time away from fully-autonomous vehicles being available to the public.

A huge factor in allowing them to be available for the average consumer is to trust that the technology driving them is working without fault, not only when around other similar vehicles with which it can communicate, but also with those vehicles being driven or controlled by a human inside the vehicle.

In his paper 'A Study of Public Acceptance of Autonomous Vehicles' Sean V Casley found that in a study of 467 people, 82.41% ranked safety as the most important aspect of autonomous vehicles compared with relevant costs and well-designed laws (Casley, 2013). These new vehicles will therefore not only have to be safe but prove that they will ensure the safety of all passengers is at a greater level than if it were to be non-autonomous. A user is as a result unlikely to increase their chances of an accident in the vehicle for the ease of use of not having to be in control. This emphasises the importance on having the effectiveness of the algorithm as a primary concern, with small faults in its functionality needing to be solved.

Pathfinder Algorithms and Methods

A large part in ensuring the safety of passengers within an autonomous vehicle comes from the technologies ability to navigate a correct path, while avoiding objects or obstructions. Pathfinding is used to dictate a computers method of navigation, using an algorithm, or series of algorithms, to determine the best route from one location to another.

In order for a pathfinding algorithm to be tested, a simulation environment needs to be created. Eka Risky Firmansyah addressed this problem by utilising a 2-dimensional array when creating a conceptual model of a 10x10 square grid, a fairly common method of representing a map in digital form. Within the array, each item serves as a single location on a map, whereby the value in each represents the type, ranging from an available or obstructed location, with special case nodes used when appropriate.

Different techniques can be used to determine the effectiveness of an algorithm. In the paper 'A comparative Analysis of A* and Basic Theta* Algorithm in Android-Based Pathfinding Games', an analogy approach is used to understand the differences in how they both solved the problem (Firmansyah, et al., 2016). Three outputs were used to measure the effectiveness of A* and Theta*, running time, path length, and nodes searched. These provide a valuable insight, showing that they both provide a similar completeness, but with A* visiting a fewer number of nodes, and with Theta* finding a

more optimal route. With A* being a common pathfinding algorithm, a comparison against various other methods would have provided a much better conclusion, with no explanation being provided as to the extent of how each one could have been better than the other.

Khammapun Khantanapoka on the other hand, used various pathfinding algorithms as a better approach to determine the effectiveness in the paper 'Pathfinding of 2D & 3D Game Real-Time Strategy with Depth Direction A*Algorithm for Multi-Layers'. Instead of using a single pathfinding algorithm for comparison, a total of seven were tested, proving the concept of how much quicker a Depth Direction A* algorithm is (Khantanapoka, et al., 2009).

Pre-generated maps were used by both, providing some possibility of maps being created to directly benefit certain approaches, where some algorithms can perform better under certain conditions. Generating random or semi-random maps could allow for an increased number of tests to be run with perhaps the ability to gradually increase the number of obstructions to see how each algorithm performs in varying conditions. Specifically with autonomous vehicles, a pathfinding algorithm will be required to constantly solve new maps, not only three or four.

Simulation Environments

Simulations can exist in many variations. In the early stages of the project, a basic grid will likely be used, similar to that which stores the map itself. 3D simulations could be implemented later, if the algorithm is proved to be effective in more basic situations.

The difficulty in simulations can often come from porting the codebase from the basic algorithm into an environment with a greater degree of complexity. Although the algorithm may be proved to work with a grid system, getting to it work outside of a programming development environment could pose issues. In the paper by Lucas Nogueira, 'A Comparative Analysis between Gazebo and V-REP Robot Simulators', he talks about the use of these programs for "testing of control algorithms for different platforms" (Nogueira, 2014). It is also mentioned how instead of programming structures for an algorithm to interact with, sensors built in to both of the robotic simulators can output information directly to the algorithm for processing.

As well as the robot instance itself, the environment has to provide important information to the algorithm. Nogueira continues his evaluation of the two methods by analysing the ease of creating a "world" (Nogueira, 2014). V-REP in particular allowed for constructing environments from scratch, with a library of models for easy addition into the current simulation. This method allows for the simplification of establishing the simulation, with a greater focus on making the algorithm work in the context, instead of the porting of logic from the early stages into a closer to real-life example.

Conclusion

There is plentiful information available for analysing pathfinding methods, however little with a direct application to autonomous vehicles, where the software is kept mostly behind closed doors by those organisations developing them. As a result, it will be interesting to see how applicable graph traversal methods are to a real-life situation.

It was found that any benchmarking performed in comparison to the new algorithm will need to be performed similarly to how Khantanapoka Khammapun and Chinnasarn Krisana compared seven different pathfinding methods. Knowing they only used the time taken and nodes expanded to, to analyse their results, additional metrics could be included for a more rounded idea of how each algorithm handled finding a path.

Instead of being targeted at a large gap in research, the project will continue a search for improving pathfinding methods. The hope is for the new algorithm to find a niche area in which it could be utilised for better optimised pathfinding, potentially finding a use within autonomous vehicles, should the testing be successful.

2.3 Bibliography (key texts for your literature review)

A Formal Basis for the Heuristic Determination of Minimum Cost Paths [Online] / auth. Peter Hart Nils J., Raphael Bertram // ieeexplore.ieee.org. - 1968. - 29 January 2017. - http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=4082035.
A Study of Public Acceptance of Autonomous Cars [Online] / auth. Casley Sean V.. - 30 April 2013. - 24 January 2017. - https://web.wpi.edu/Pubs/E-project/Available/E-project-043013-155601/unrestricted/A_Study_of_Public_Acceptance_of_Autonomous_Cars.pdf.
Company History [Online] / auth. Daimler // daimler.com. - 26 January 2017. - https://www.daimler.com/company/tradition/company-history/1885-1886.html.
Comparative Analysis Between Gazebo and V-REP Robotic Simulators [Online] / auth. Nogueira Lucas // http://www.dca.fee.unicamp.br. - 2014. - 2018. - http://www.dca.fee.unicamp.br/~gudwin/courses/IA889/2014/IA889-02.pdf.
Comparitive Analysis of A* and Basic Theta* Algorithm in Android-Based Pathfinding Games [Online] / auth. Firmansyah Eka Risky, Ummi Masruroh Siti and Fahrianto Feri // ieeexplore.ieee.org. - 2016. - 2017. - http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7814916.
Despite CES hype, self-driving cars are not for sale [Online] / auth. Warren Tamara // The Verge. - 13 January 2018. - 26 January 2017. - https://www.theverge.com/2018/1/13/16887728/detroit-auto-show-2018-ces-self-driving-cars-toyota-ford-hype.
Jaguar Land Rover tests driverless cars on public roads [Online] / auth. BBC // BBC News. - 17 November 2017. - 27 January 2017. - Jaguar Land Rover tests driverless cars on public roads.
Pathfinding of 2D & 3D Game Real-Time Strategy with Depth Direction A*Algorithm for Multi-Layer [Online] / auth. Khantanapoka Khammapun and Chinnasarn Krisana // ieeexplore.ieee.org. - 2009. - 2017. - http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5340922.

THIS IS THE END OF SECTION TWO

Appendix B - Project Presentation

The Project So Far...

Algorithms

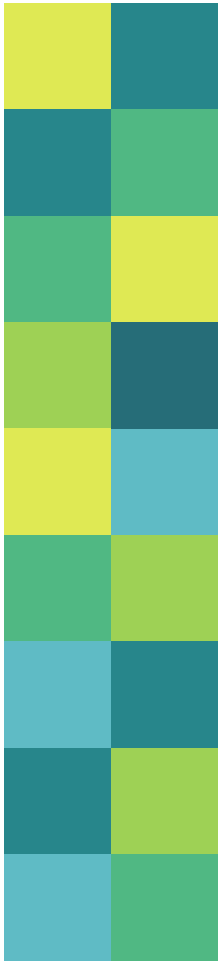
- New algorithm has been mostly implemented. Issues arose with object avoidance.
- Existing algorithms have been implemented but need converting to work with project.

Testing

- Module for performing the tests needs to be written. Will contain start() and finish() functions to call the algorithms to be tested.
- Storage of data requires some thought.

Write-Up

- Full plan in place giving an idea for how it will come together.



Demonstration.

To Complete the Project...

Algorithms

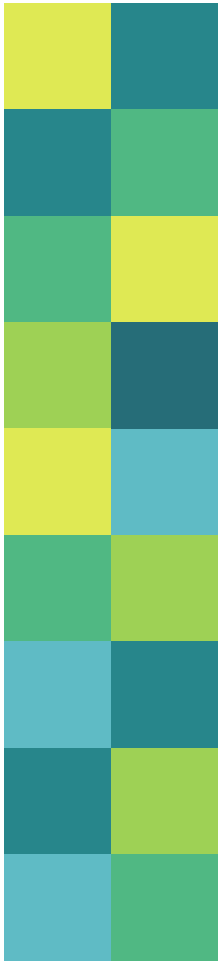
- Continuing development with the new algorithm in its final stages.
- Existing algorithms need some modifications to finish at specific end point.

Testing

- Both Profile and Time libraries will be used for analysing the effectiveness.
- Time will contain Python overhead, however profile is more difficult for data extraction.

Write-Up

- Each chapter will require more in-depth sub titles for what to mention.
- Want these to be added as sections of the program are written to cover all elements.



Appendix C - Meeting Records

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 24/11/2017

Topics Discussed:

- General project ideas
- Areas of interest
- Pathfinding?

Identification of Issues:

Actions Set Before Next Meeting:

- Research pathfinding as a concept
- Community research

Date of Next Meeting:	2 weeks time

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 07/12/2017

Topics Discussed:

- Direction of project
- Pathfinding → new algorithm?
- Data analysis → care home data
  - machine learning

Identification of Issues:

- Little knowledge surrounding machine learning
- How to write proposal

Actions Set Before Next Meeting:

- Pick topic → likely to be pathfinding-based
- More research into that area
  - Papers, not just websites
- Start proposal

Date of Next Meeting:	First week back from Christmas

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: ~~Euan~~

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 24/01/2018

Topics Discussed:

- Pathfinding -> chosen for project
- Look at first draft
- Idea development
  - Where to go with it
- Realistic scope

Identification of Issues:

- Scope can be narrowed down to focus more on algorithm than comparisons.

Actions Set Before Next Meeting:

- Continue to improve proposal
- Make changes based on feedback given

Date of Next Meeting:	Next week - 29th/30th Jan

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: 

Supervisor: **Yanguo Jing** Supervisor Signature: 

Date: 30/01/2018

Topics Discussed:

- Discussion of changes to proposal

Identification of Issues:

- Issues identified in document

Actions Set Before Next Meeting:

- Make changes, continue improvements

Date of Next Meeting:	Next week - before deadline

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 07/02/2018

Topics Discussed:

- Discussion of changes to proposal

Identification of Issues:

- Any issues listed with document.

Actions Set Before Next Meeting:

- Finish and submit proposal

Date of Next Meeting:	2 weeks time, After proposal marked.

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: 

Supervisor: **Yanguo Jing** Supervisor Signature: 

Date: 22/02/2018

Topics Discussed:

- Explanation of algorithm methodology
- Look at the data structure supporting the algorithm

Identification of Issues:

- Research is important, don't jump into practical with little understanding of area.

Actions Set Before Next Meeting:

- More detailed literature review plan
- Map generation programmed.
  - Will require research
  - Look for existing methods

Date of Next Meeting:	Week commencing 26th March

Snow Week

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record (Email feedback)

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 01/03/2018

Topics Discussed:

- Structuring the program
- Explanation of map generation
- Sent GitHub link

Identification of Issues:

—

Actions Set Before Next Meeting:

- Build algorithm handler

Date of Next Meeting:	Next week - Commencing 5th March

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 07/03/2018

Topics Discussed:

- Collecting resources for literature review
- Algorithm development

Identification of Issues:

- Algorithm taking longer than expected to work
  - Object avoidance issues
- Unlikely to reach scope 2 or 3.

Actions Set Before Next Meeting:

- Try and get algorithm at a "working" state

Date of Next Meeting:	——

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record (Email feedback)

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 16/03/2018

Topics Discussed:

- Can start to test algorithms (comparison) work
- Converter built to change map format
  - For graph-based algorithms
- Algorithm development

Identification of Issues:

Actions Set Before Next Meeting:

- Integration of MatplotLib for visual representation.

Date of Next Meeting:	21/03/2018

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

(New Wizards Showcase)

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 21/03/2018

Topics Discussed:

- Testing methodology
- Showed algorithm working

Identification of Issues:

- Need better object avoidance
- Does not work in all situations

Actions Set Before Next Meeting:

- Improve algorithm, needs to be working for testing soon.

Date of Next Meeting:	Next week - 26th March

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record (Presentation)

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 28/03/2018

Topics Discussed:

- Stage of development
- Testing methodology
- Write up
- Initial / Actual testing?
- New areas for pathfinding
  - not just navigation
- Comparison algorithms
  - could be vary depending on implementation

Identification of Issues:

Actions Set Before Next Meeting:

- Finish algorithm
- Continue write up for early stages
- Perform testing / analysis
  - Write it up

Date of Next Meeting:	After Easter break

An investigation of pathfinding for navigating autonomous vehicles

Meeting Record

Student: **Euan Campbell (6384171)** Student Signature: Euan

Supervisor: **Yanguo Jing** Supervisor Signature:

Date: 26/04/2018

Topics Discussed:

- Update on testing performed
- Discussion of results
  - What they mean
  - how to talk about them
- Write up

Identification of Issues:

- All images need to be labelled
- Chapters need adjusted titles
- Conclusion doesn't reflect objectives
- No ethical considerations

Actions Set Before Next Meeting:

- Make changes to write up
- Email Yanguo final draft
- SUBMIT

Date of Next Meeting:	

Appendix D - Certificate of Ethical Approval

Certificate of Ethical Approval

Applicant:

Euan Campbell

Project Title:

An investigation of pathfinding for navigating autonomous vehicles.

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:

22 February 2018

Project Reference Number:

P67726