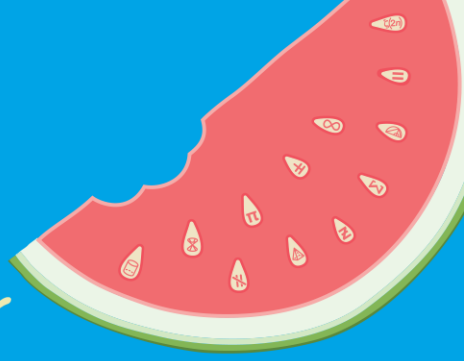
AMSI VACATIONRESEARCH SCHOLARSHIPS 2021–22

Get a taste for Research this Summer



Cellular Automata Traffic Network Analysis

Jamie Keegan-Treloar

Supervised by: Vladimir Ejov & Iwan Jensen

Flinders University

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Contents

1 Abstract

Intersection selection is a complex task. The traffic volume and use case must be considered. This project creates an interactive simulation of the three main intersection types in an attempt to make traffic analysis more accessible. The simulation uses a mix of cellular automata for the network layout and object oriented programming for the car and intersection behaviour. A network is then created and various traffic volumes populated into it to test the long-term behaviour of each intersection. It is found each intersection performs optimally for different traffic volumes. This illustrates the complexity of road network design.

2 Introduction

With the population size and density growing rapidly, optimal road networks are becoming more vital than ever before. A well designed road network can reduce travel-times, emissions and much more. Selecting the right intersection type *(Give-way, Traffic-Lights or Roundabout)* is the area of interest for road network optimisation. Each intersection types has its own advantages and disadvantages with optimal performance at different traffic volumes. This project creates a Cellular Automata traffic simulation of the three intersection types and analyses how each performs, ultimately providing insight into road network design and where each intersection may be most appropriate.

3 Statement of Authorship

This project was written by Jamie Keegan-Treloar. The main idea was proposed by Jamie Keegan-Treloar with the idea of cellular automata proposed by Iwan Jensen. There were meetings throughout where Vladimir Ejov and Iwan Jensen suggested changes to the simulation which was fully developed by Jamie Keegan-Treloar.

4 Project Goal

Traffic analysis is an inaccessible and highly technical area of study with the need for complex programs and significant computation power. This project aims to bridge this divide by creating an accessible, intuitive, interactive and rigorous method for simulating traffic behaviour. The program should be hosted on a web server with an interactive user-interface. Hopefully this interactive program could be used as both a tool and a resource to highlight the everyday uses for mathematics.

5 Cellular Automata

Cellular Automata *(CA)* is a set of cells on a specific grid which evolve over discrete time steps according to a a rule-set [4]. The behaviour of an individual cell is dependant on both the rule-set and its neighbouring cells. CA has been used for simulating behaviour of many complex systems, including the behaviour of organisms and how gases propogate [4]. However, the most famous example of CA would be Conway's Game of Life where cells evolve depending on the resources and space available [1].

In Conway's Game of Life, the rule-set is very simple:

- If the cell is alive and has 2 or 3 neighbours it remains alive
  - else it dies
- If a dead cell has exactly 3 alive neighbours it becomes alive

From this very simple rule-set, complex behaviour can arise with certain configurations being able to move [1] or even be used as binary logic gates [3].

For this project, the grid will be the road network with a cell representing a small section of the road. This cell can either be occupied or empty (alive or dead), the cell will have unique attributes such as destination and speed. Using this additional layer, more complex situations can be modelled.

6 Project Structure

Traffic is a very complex process with many interacting objects and situation dependant rules. To make this project achievable in the brief time-frame that is the AMSI VRS, it had to be broken into manageable tasks with a clear goal. The main element of the project would be the development of the simulation, building the rule-sets to incorporate all the intersection types. The structure of the development is as follows:

- Develop basic move and queue behaviour
- Add roundabout rules
- Add give way rules
- Add traffic light rules
- Use the program to analyse traffic

7 Rule-Sets

A rule-set defines the behaviour of the grid and subsequently must be realistic for the situation. Many assumptions must be made to break the real-world problem into a simulation, these assumptions will be stated.

7.1 Grid Foundations

Before the rule-sets can be implemented, there must be a defined grid. As this program is interactive, a simple user interface *(UI)* was created where users could click the mouse to place intersections and then draw the roads between, ultimately forming a unique grid layout.

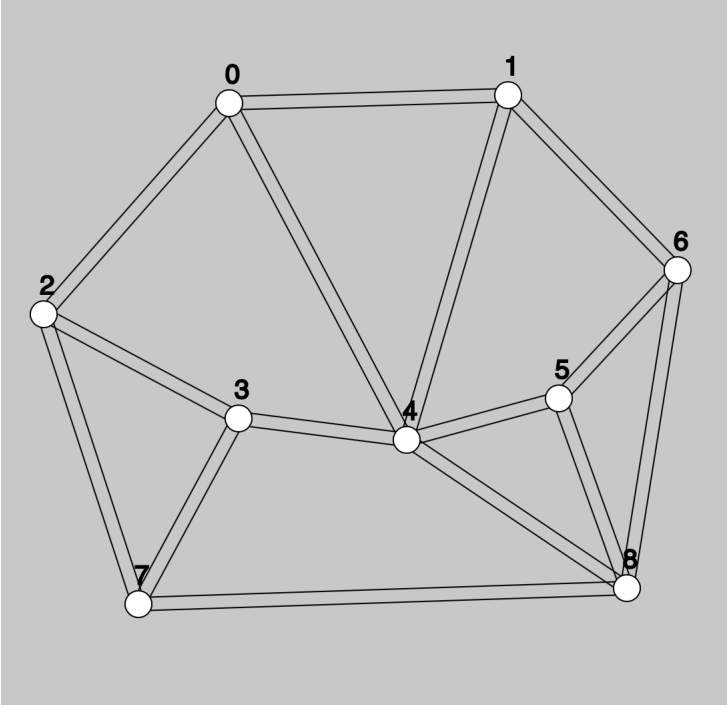


Figure 1: Grid created using the mouse and the road connections method

As shown in figure 1, very complex networks can be created.

Both the intersections and the roads are objects which can have unique attributes. These attributes are listed below.

Intersections	**Roads**
Type	Start
Position	End
Connected Roads	Speed
Connected Intersections	Traffic
ID	ID

Table 1: Attributes for intersection and road classes

These attributes allow for complex operations which will be used later.

To save on computation, cars in the cells can be made into objects. These car objects, either occupying a cell or not, have the following attributes.

Car
Position
Target
Speed
Previous Targets
ID

Table 2: List of car object attributes

Combining the grid-like simplification with the complexity of objects will allow for more complex and adjustable behaviour.

7.2 Queuing and Movement Behaviour

The fundamental rule-set is how the alive cells move and queue when there is no room. This is done in a very simply function.

- Loop all car objects
  - Create a vector heading from the car to the target
    - If there is a car or intersection within a set distance in the direction of the heading:
      - Don't move
    - else:
      - Move a set number of cells in the direction of the target

For this first function the following assumptions are made.

- Cars have the same max speed of 1
- Cars react instantly (no response time)
- Cars accelerate instantly
- Cars which can't move have a speed of 0

The queuing and move function worked correctly allowing for the addition of basic intersection turning behaviour.

7.3 Roundabout Behaviour

The rule-set for a roundabout is relatively simple. By following through the logical steps involved in the road rules the process can be converted into a function.

The steps involved for safely turning are as follows.

- Approach the roundabout adhering to the queuing function from before
- If the car has arrived at the roundabout then search for all cars within a set radius of the intersection
  - Create a vector from each car to the roundabout
  - Take the heading of each vector
    - Check whether any car has a heading within a set alpha placing the car to the right (if there is a car, it is within the distance and to the right)
    - If there is a car don't move
    - If there is not a car then move
      - Update cars attributes with new target intersection

By adjusting the search radius around the roundabout and the alpha value for vector headings, the roundabout can be customised to match real world situations.

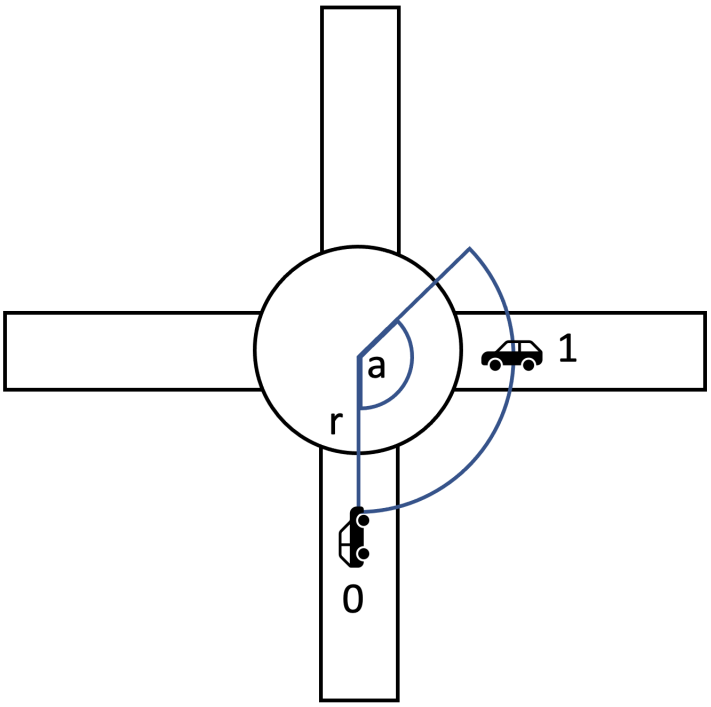


Figure 2: Diagram showing the adjustable search radius and alpha value for roundabouts

Figure 2 shows the adjustable region the car object (going upward, ID: 0) checks to give way to or proceed. In this figure, there is a car (ID: 1) within this region which would then cause car 0 to give way. This is a realistic replication of the real-world process (for left hand driving countries) and performed appropriately in the tests.

7.4 Give Way Behaviour

The give way rule-set is an extension to the move and queuing function. In a give way intersection there is a dominant and non-dominant flow. The behaviour for each direction is dependant on the origin (dominant or not) and which flows it must cross. A list or the roads is created, sorted by the angle of the road to the intersection and then re-indexed to assert the first item is a dominant flow. This allows for the give way instructions to be easily found from a generic lookup table. The function works as follows.

- Create a list of the angles formed from the intersection to the road
- Sort the list by the angles
- Rotate the list until the first index is a dominant flow
- Find the indexes of the cars entry and exit road in the sorted list
- Compare these values against a generic lookup table

It should be noted that a single lookup table can be created for a four-way intersection and used for a three-way intersection by inserting a null road between the two dominant roads creating a cross road. The program includes a sub-menu for selecting the dominant flow.

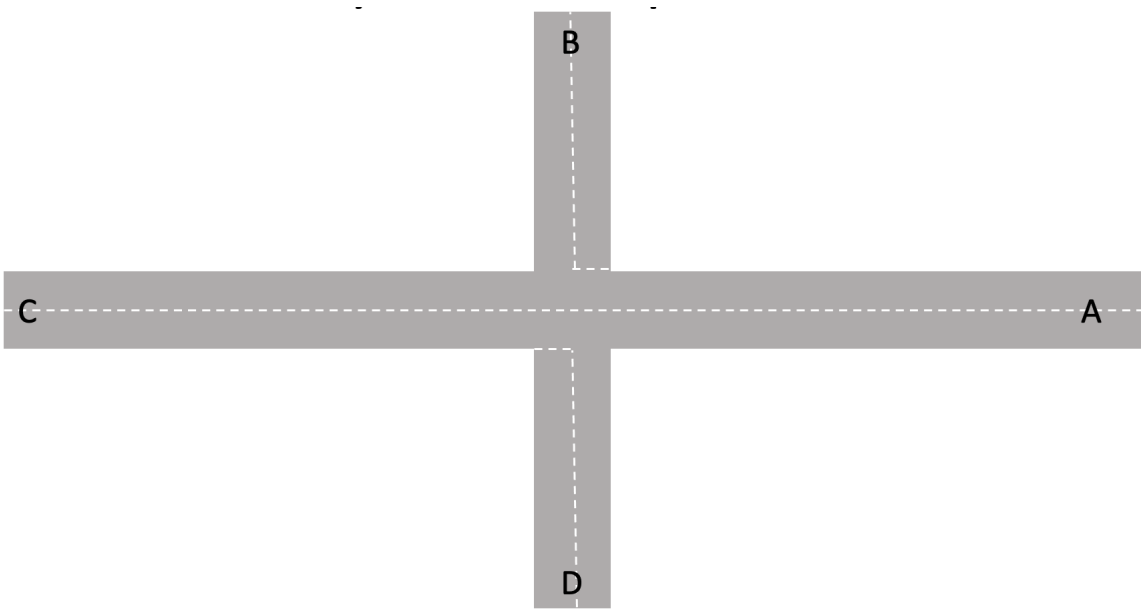


Figure 3: Cross road with the entries and exits labelled. The flow A-C is domiant.

Figure 3 shows how ordering the roads by the angle created to the intersection can then allow the lists indexes to be used to replace the letters. This diagram can be used to construct the lookup table.

FROM \ TO	A	B	C	D
A		C-A, C-D, C-B		
B	C-A		A-C, A-B, C-A, C-D, D-B, D-C	C-A, C-D, A-B, A-C, A-D
C				A-D, A-B, A-C
D	C-A, C-D, B-A, A-C, A-B, B-D	A-B, A-C, C-B, C-A, C-D	A-C	

Figure 4: Lookup table for a cross road give way intersection showing the paths each flow must give way to

The lookup table, shown in figure 4 is the generic table used in the program. It returns a set of lists of entries and exits to which a particular flow must give way. The current traffic in the intersections proximity is then checked against this list and if no cars have these values for the origin and destination, then the car may proceed.

The checking function works as follows.

- Create a list of cars within a certain radius of the intersection
- Find the closest car to the intersection
- Find the cars entry and exit road
- Compare the entry and exit roads to the lookup table and receive a list of entries and exits to give way to
- If any car in the list of proximity have these entries and exits don't move
- Else, move

The give way function worked correctly with reasonable flow rates and congestion levels.

7.5 Traffic Light Rule-Set

The traffic light rule-set is the most basic of the lot. A phase dictates whether a direction can move. By creating a list of flows which can move concurrently and cycling through this list, realistic traffic light behaviour can be

simulated. At each time-step of the simulation, a phase count is increased, this is used to loop the list with each phase having an adjustable length. The timing of the phases is a massive area of study and hence for simplicity, all phases have the same length.

8 Final Program

The final program is available online [2]. Having built up the program in a series of rule-sets it was finally complete. While there were still many aesthetic problems, these were not detrimental to the operation. The final program was interactive, intuitive and mathematically rigorous. The user can change the programs mode by pressing the keys.

- 'N' New Intersection Mode, allows for new intersections to be placed at the mouse position
- 'C' Connection Mode, allows for intersections to be selected by clicking on them and once two are selected a road is placed between them
- 'M' Add Cars Mode, allows for 10 cars to be randomly placed into the intersection
- 'T' Intersection Type Mode, allows for editing of selected intersections type

The user interface gives information about the current state of the network and mode.

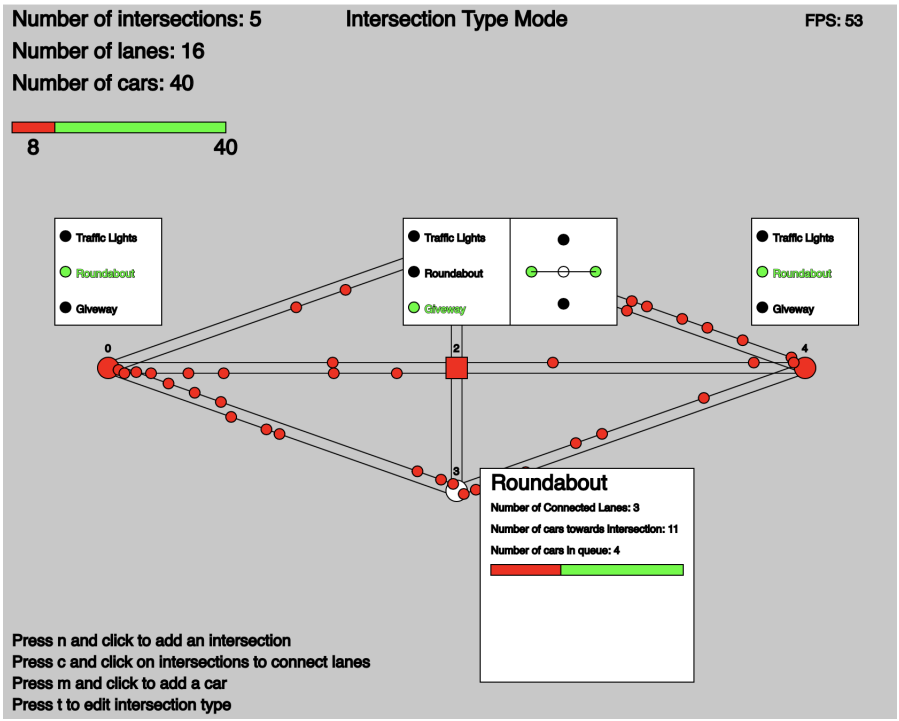


Figure 5: Interface for the program showing the network, intersection types and car speed information

Figure 5 shows the user interface of the program. The car objects are shown as small red dots and the intersections as the larger dots (roundabouts) and squares (traffic lights and give way). Above the center give way intersection, the sub-menu for selecting the dominant flow can be observed. The top left of the UI provides information on the total number of cars and also the number of cars currently slower than the maximum speed. Overall, the UI is clear and intuitive while being able to be run on a web-browser. This goal of the project has been met.

9 Analysis

While the program is running correctly, to analyse traffic, a performance metric must be selected. Using the performance metric will then provide insight into the specific area of interest, ultimately shedding light on traffic behaviour on simple road networks.

9.1 Performance Metric

There are many statistics that could be measured at an intersection; throughput (cars per minute or people per minute), evenness of speeds, evenness of flows, optimal usage of the network and much more. While each of these metrics would provide insight into a different area of traffic behaviour, the limited time-frame of this project meant only one performance metric could be measured. The chosen metric was the overall average speed of the cars in the network. As speed is either 1 (full-speed) or 0 (stopped) the metric would be on a scale from 0 to 1. This metric would provide insight at a macro level on the intersections performance.

9.2 The Analysis Setup

The analysis would be conducted on an identical and repeatable network with the same number of cars and the same number of data points. The basic network was a cross-road, with 1 intersection in the middle and the intersections on the edge turning the cars around with no delay. The central intersection would be changed to each of the three types and the analysis repeated. Over a period of 10,000 time-steps the average speed of the cars was recorded. After which, the analysis was repeated with 10 more cars. This process was conducted until either there was 160 cars or the intersection had clogged.

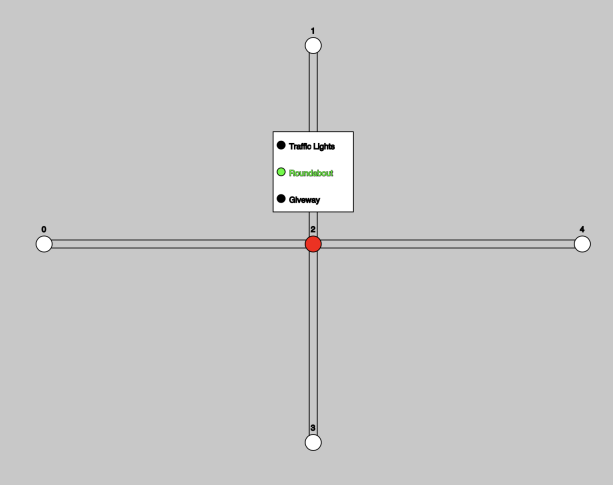


Figure 6: Test network showing the central intersection of concern

By using a repeatable network setup like in figure 6, the data collected for each intersection and car volume would be comparable.

9.3 Results

The average speed of the cars in the network followed an expected pattern. As the volume increased, the average speed decreased.

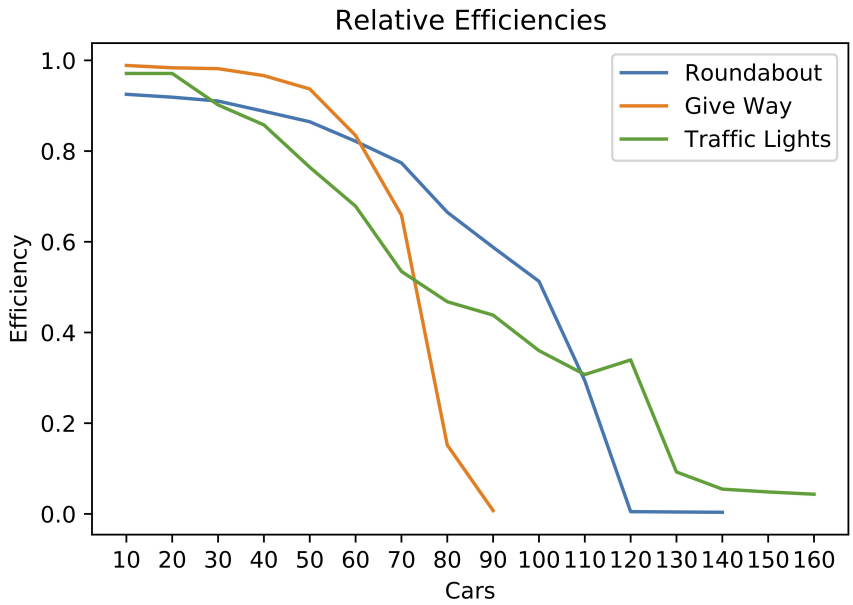


Figure 7: Relative performances of the three intersection types given traffic volume

As shown in figure 7, the general trend is downward for all intersections. The rate that the intersections slow,

varies with a rapid decrease for give way intersections, followed by the roundabout and lastly the traffic lights. The performance of the give way intersection appears close to optimal for low volumes of traffic. There are volumes where each intersection performs better than other others. These points are at around 60 cars where a roundabout becomes better than give way and 110 cars where traffic lights then become the best choice. All the intersections also cease functioning at a certain volume, again following the same order, give way, roundabout and finally traffic lights.

Figure 7 is constructed by averaging the speeds of the volumes for each intersection and therefore doesn't provide deep insight into why and how the intersections clog and the performance degrades. More insight can be provided by examining the degradation of average speed over time for a select set of volumes.

To better illustrate the oscillation and balanced state, a larger network was created to allow the cars to dissipate and equalise the spacing. The setup was similar to the previous layout, only with longer roads, allowing for more cars to be tested.

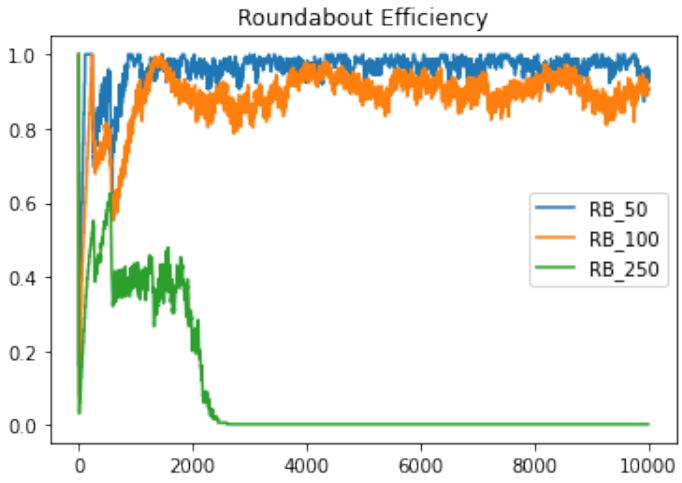


Figure 8: Degradation of roundabout average speed for traffic volumes of 50, 100 and 250 cars

When the cars initially enter the network, there are periods of queue oscillation where cars queue for long periods of time, however, this settles as the cars naturally tend to a balanced state. Figure 8 illustrates this oscillation and how larger volumes cause larger fluctuations. For roundabouts the fluctuations settle quickly highlighting the even flow behaviour of a roundabout.

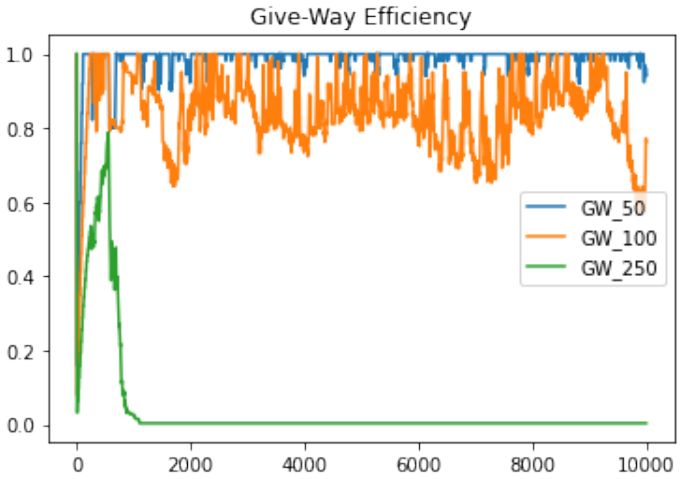


Figure 9: Degradation of give way average speed for traffic volumes of 50, 100 and 250 cars

The fluctuations caused by a give way intersection are far more drastic. Figure 9 illustrates, the previously seen, excellent performance for low volumes (shown in blue) however, with the increase of traffic volume, the fluctuations become much more frequent. This is to be expected as there is a dominant flow causing large queues on the other directions.

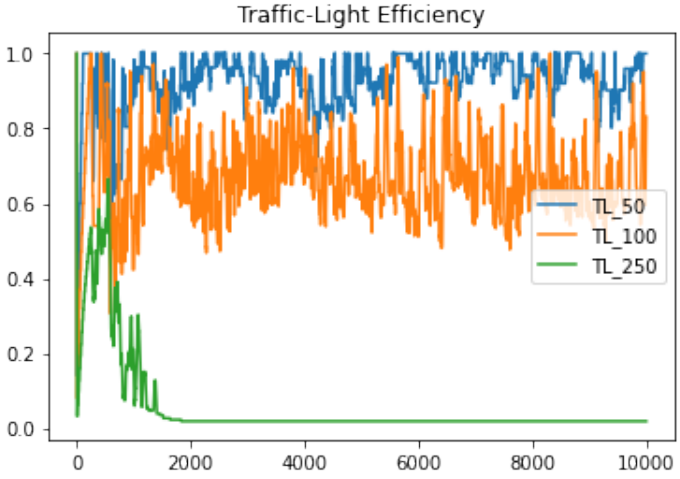


Figure 10: Degradation of traffic light average speed for traffic volumes of 50, 100 and 250 cars

Lastly, traffic lights cause frequent fluctuations, even for low volumes. This is expected due to how the lights stop all directions causing many queues.

From the plots of performance over time, it can be observed that roundabouts maintain even queue periods with no large fluctuations arising from a dominant flow. While give way performs optimally at low traffic volumes this performance quickly declines. The traffic lights may stop many directions and cause large fluctuations of

speed and subsequently queues but these queues are even and remain relatively constant over time.

10 Discussion

Using the combination of cellular automata and object oriented programming to make this simulation provided insight into the advantages and use case of each intersection type. Choosing the performance metric to measure is a large part of the analysis. While this project only measured the average speed, in the future it would be interesting to implement and test other metrics. While, initially, it may appear a roundabout or give way would perform better, there are traffic volumes which influence this choice. It was observed that give way performed best for low traffic volumes. Followed by the roundabout and lastly traffic lights. This project illustrated how each intersection should be selected based on the possible traffic volumes.

The three intersection types clogged at different volumes for various reasons. Clogging occurs due to cars giving way to another car which in turn is giving way. This cycles around the intersection resulting in no movement. In later revisions of the program a jam mode might be added to account for peak-hour traffic. Other changes to the code may include more variation for driver behaviour to better mimic real-world drivers.

11 The Website

To access the website please visit: https://docgunter.github.io/AMSI/

12 Conclusion

This project created a cellular automata and object oriented traffic simulation in the hopes of making traffic analysis more accessible. The simulation was built-up in a series of rule-sets until all intersection types and queuing behaviour was being simulated. The program can now be hosted on a website and used by anyone with a computer. This makes it both accessible and interactive, meaning people can create their own networks and observe the traffic behaviour. This simulation was then used to do simple analysis of the three intersection types to determine how each performs. It was found for low traffic volumes, a give way intersection is best. However, as the traffic volume increases, roundabouts perform best then later traffic lights. Choosing an intersection for any particular road network is a complex task. The traffic volume must be predicted. Hopefully, with this simulation, more people can take interested in and appreciate the road networks around us.

References

[1] URL: http://pi.math.cornell.edu/~lipa/mec/lesson6.html.

[2] URL: https://docgunter.github.io/AMSI/.

[3] Paul Rendell. "A Universal Turing Machine in Conway's Game of Life". In: *2011 International Conference on High Performance Computing Simulation*. 2011, pp. 764–772. DOI: 10.1109/HPCSim.2011.5999906.

[4] Eric W. Weisstein. *Cellular Automaton*. URL: https://mathworld.wolfram.com/.