Traffic Lights Optimization using Simulation: Amber before Green Scenario in NSW

Yamit Chinchilla Ospina

22026488

A report submitted for

INFO7017 Postgraduate Project B

in partial fulfillment of the requirements for the degree of

Master of Data Science

Supervisor: Oliver Obst

School of Computer, Data and Mathematical Sciences
Western Sydney University

2023

ABSTRACT

Over the course of many years, the rising transportation demands and urban road traffic have given rise to a multitude of challenges in urban areas. These issues include traffic congestion, delays, lengthy queues, increased travel times, road accidents and environmental concerns. Urban Traffic Control has made it a priority to address these issues by increasing traffic flow along specified routes, making it a crucial strategy for reducing traffic issues and enhancing mobility in urban area. In this detailed report, we present a comprehensive traffic simulation model designed to emulate real-world urban traffic scenarios accurately. The simulation incorporates sophisticated traffic patterns, signal timings, vehicle behaviors, and essential metrics to create a realistic intersection management system.

The key tool to our simulation is the Traffic Light Control Logic. Utilizing a multi-phase traffic signal system, we dynamically manage signal timings based on vehicle counts and traffic flow, optimizing the intersection's efficiency. The simulation includes vehicle generation, movement, and the integration of start-up lost time, representing the interval between signal changes and the moment vehicles begin moving. This report provides insights into the algorithms, data structures, and simulation logic employed. We explore vehicle categorization, accounting for diverse behaviors such as turning, acceleration, and deceleration. The simulation's accuracy and adaptability are demonstrated through various scenarios, highlighting its ability to handle changing traffic conditions effectively.

Our study emphasizes the importance of start-up lost time, perception reaction time and Idle time in the intersection critical parameters affecting traffic flow dynamics. By integrating this factor, our simulation captures the delay between signal changes and vehicle movement, enhancing the model's realism. Throughout the report, we discuss the system's robustness, flexibility, and potential applications in urban planning and traffic management. While collision detection remains a future area of exploration, the current simulation framework serves as a foundational tool for understanding traffic dynamics and optimizing signal control strategies.

ACKNOWLEDGMENTS

I want to express our heartfelt gratitude to my parents for their unwavering, constant support, encouragement, and understanding throughout this project. Their belief in my abilities has been a constant source of motivation, and their support has been invaluable in every step of the way.

We are deeply thankful to our supervisor, **Oliver Obst** , for his expert guidance, insightful feedback, and unwavering encouragement. His mentorship significantly shaped the direction of our research and enriched our understanding of the subject. Additionally, I want to extend my appreciation to the participants who generously shared their time and experiences, enriching my study with real-world perspectives.

To our colleagues and friends, we value your camaraderie and the stimulating discussions that provided fresh insights and ideas, enhancing the quality of our work.

Lastly, we acknowledge the academic community for their scholarly contributions, which laid the foundation for our research.

The collective encouragement and support from our parents, Supervisor Oliver Obst, participants, colleagues, and the academic community have been instrumental in the successful completion of this project. Thank you for your invaluable contributions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. CHAPTER I: INTRODUCTION

In an era marked by rapid urbanization and increasing vehicular traffic, efficient traffic management systems are imperative to ensure the smooth flow of vehicles, enhance road safety, and reduce congestion. As urban centers expand, the need for intelligent traffic control systems becomes paramount. This project delves into the development and simulation of a sophisticated traffic signal control system, a fundamental component of intelligent traffic management.

Traditional traffic signal systems follow predefined schedules, often leading to inefficiencies during varying traffic conditions. This project aims to address these challenges by implementing a dynamic traffic signal control system that adapts to real-time traffic patterns. The objective is to optimize traffic flow, minimize waiting times at intersections, and enhance overall road safety.

The primary focus of this study revolves around the simulation and analysis of traffic behavior at intersections under different signal control strategies. Through the integration of advanced algorithms and simulation techniques, we explore innovative methods to improve traffic signal synchronization, reduce startup delays, and optimize signal timings. Additionally, the project investigates the impact of these strategies on vehicular movement, idle times, and overall intersection efficiency.

By leveraging simulation software and computational models, this research endeavors to provide valuable insights into the performance of adaptive traffic signal control systems. Understanding these dynamics can inform real-world implementations, leading to more intelligent, responsive, and efficient traffic management solutions. Through this project, we contribute to the ongoing discourse on smart urban transportation, aiming to create safer, greener, and more accessible cities for all.

2. CHAPTER II: Literature Review

Over the course of many years, the rising transportation demands and urban road traffic have given rise to a multitude of challenges in urban areas. These issues include traffic congestion, delays, lengthy queues, increased travel times, road accidents and environmental concerns. Urban Traffic Control has made it a priority to address these issues by increasing traffic flow along specified routes, making it a crucial strategy for reducing traffic issues and enhancing mobility in urban areas. Two popular methods for improving urban transportation are the "Transition Phase" and "Traffic Light Optimisation." The time when changes to traffic-timing plans are made is known as the "Transition Phase". These modifications, which aim to restore coordination with the newly developed plan, may include parameters like offset, green time and cycle duration (Bouktif, 2023). On the other hand, "Traffic Light Optimisation" focuses on enhancing mobility through the application of mathematical models and other techniques like parameter estimation methods. One straightforward method employs a microscopic traffic representation, emphasising the examination of individual vehicle behaviours within the road network, including interactions among drivers, vehicles and infrastructure.

In addition, algorithms for traffic optimisation have been presented that may adaptively control traffic and produce the best results. The creation of more effective algorithms has drawn increased attention to ideas like parameter estimation, fuzzy logic and artificial intelligence (Ding, 2022), (Ducrocq, 2023). The invention of a stochastic traffic signal light timing optimisation model, intended to account for daily variations in delay during the optimisation process, is a noteworthy development. Quantifying the variance in delay at signalised junctions and seamlessly incorporating it into the optimisation process were the main goals of this study (Hong, 2022). The three categories of traffic signal control strategies are actuated control, adaptive control and fixed time control. The rationale for this classification lies in the algorithms they employ for optimisation.

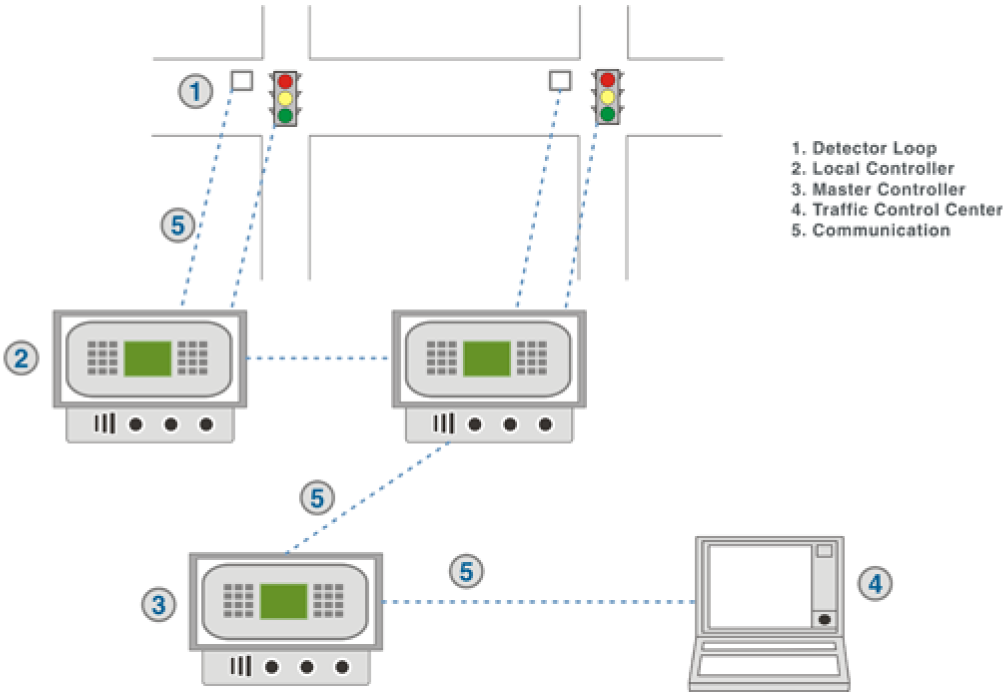


Figure 1: Actual Physical Parameters of traffic signal

When delving into research related to traffic signal management, two broad categories emerge research using "computational intelligence models" that combine machine learning and optimisation approaches, as well as studies using microsimulation methods. These strategies cover a broad range and include AI-based methodologies, metaheuristic methods, multifaceted tactics, neural networks, fuzzy models, swarm intelligence and bi-level programming are some examples of machine learning techniques (Kolat, 2023). For example, Turky's strategy (Korecki, 2023) solves the drawbacks of conventional traffic controllers by efficiently accounting for dynamic traffic load inputs as the main idea. To track the number of vehicles and people at the road junction, this creative solution uses sensors installed at each lane of a four-way, two-lane junction (Ducrocq, 2023). Another noteworthy instance is Sanchez's model (Lin, 2022), which determined that the absolute number of vehicles leaving the network was the best performance metric for traffic simulation. This metric takes into account how quickly both incoming and departing cars get to their destinations, comparisons with different scenarios are made easier thanks to this detailed evaluation of simulation efficacy (Liu, 2022). In essence, these systems bear similarities to those currently employed in Sydney, which offer real-time information monitoring, alarms, access to historical data and user-friendly interfaces.

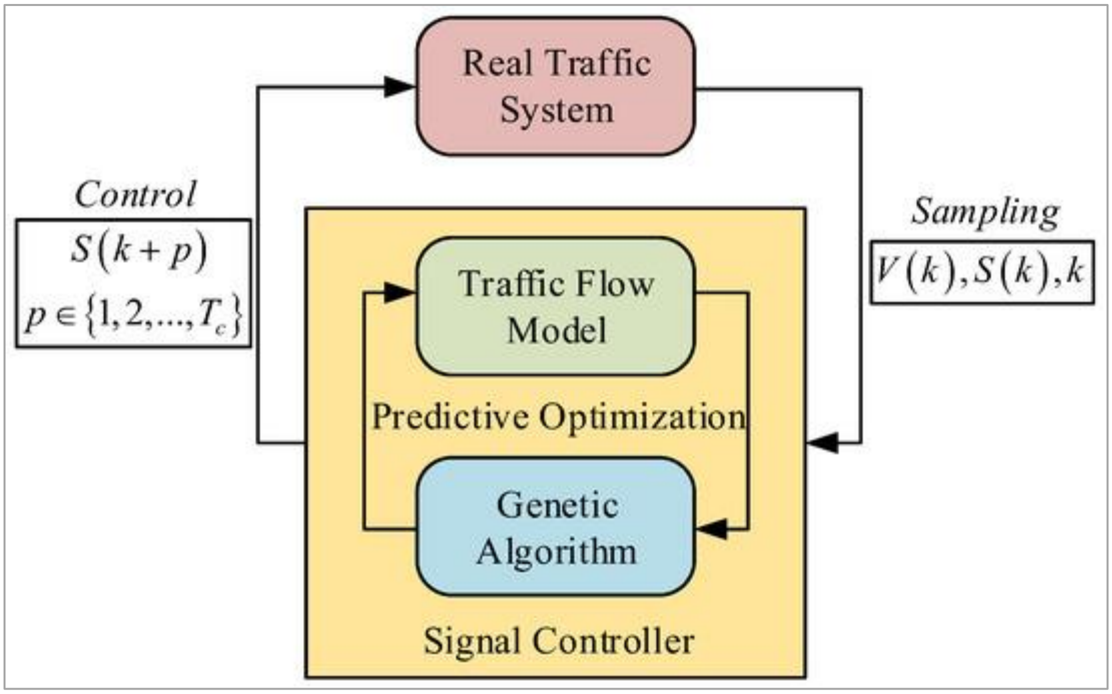


Figure 2: Traffic Flow Model Application of delay timer

Creating an accurate picture of traffic that corresponds with real observations and measurements on the street is the main goal of traffic modelling. Traffic modelling relies on the skills of modellers to smoothly incorporate mathematical models into the system, as opposed to merely mimicking the actual appearance of the traffic system. Case studies, like that of Abilene in Texas, USA, have shown that light traffic optimisation not only improves traffic flow throughout the network but also results in significant savings on fuel and CO and HC emissions.

1.1Historical Context of Traffic Lights

In the early 1900s, The first traffic light system, also known as "Morgan's Safety Hooded Traffic Control," was created by African-American inventor and businessman Garrett Morgan. A red light, a green light and a white warning light were put on a T-shaped pole in this novel design. The green light meant that traffic could move forward, while the red light was a clear indication to "Stop" traffic in all directions. Notably, Morgan's original design did not include the modern-day amber light. Instead, it employed a white warning light as a stand-in signal to warn motorists to proceed with care and yield before the red light turned on. Taking this extra stride made it possible for pedestrians to safely cross the crossing. William Potts later popularised the idea of using an amber light to mark a change from red to green in the United States. This innovation aimed to create a smoother transition for drivers, promoting improved traffic safety at intersections.

1.1.1 Traffic Light Phases and Sequences

Traffic signal controls are integral in managing traffic flow at intersections and major roadways. They allocate specific time intervals to different movements to enhance traffic flow and efficiency. The typical sequence of traffic light phases consists of:

Red: Signifying a complete stop for all traffic.

Red + Yellow: Indicating the impending change from red to green, prompting drivers to prepare for movement.

Green: Allowing traffic to move forward.

Yellow/Amber: Signaling a transition from green to red, prompting drivers to slow down and prepare to stop.

Research focused on optimising traffic light sequences, including the use of an amber light, is the investigation that Schutter and Moore suggest (Qadri, 2020). The amber light was introduced into this model to examine queue growth and congestion. The goal of the study was to identify a red-amber-green cycle with changes that would produce the best sequence of lights.

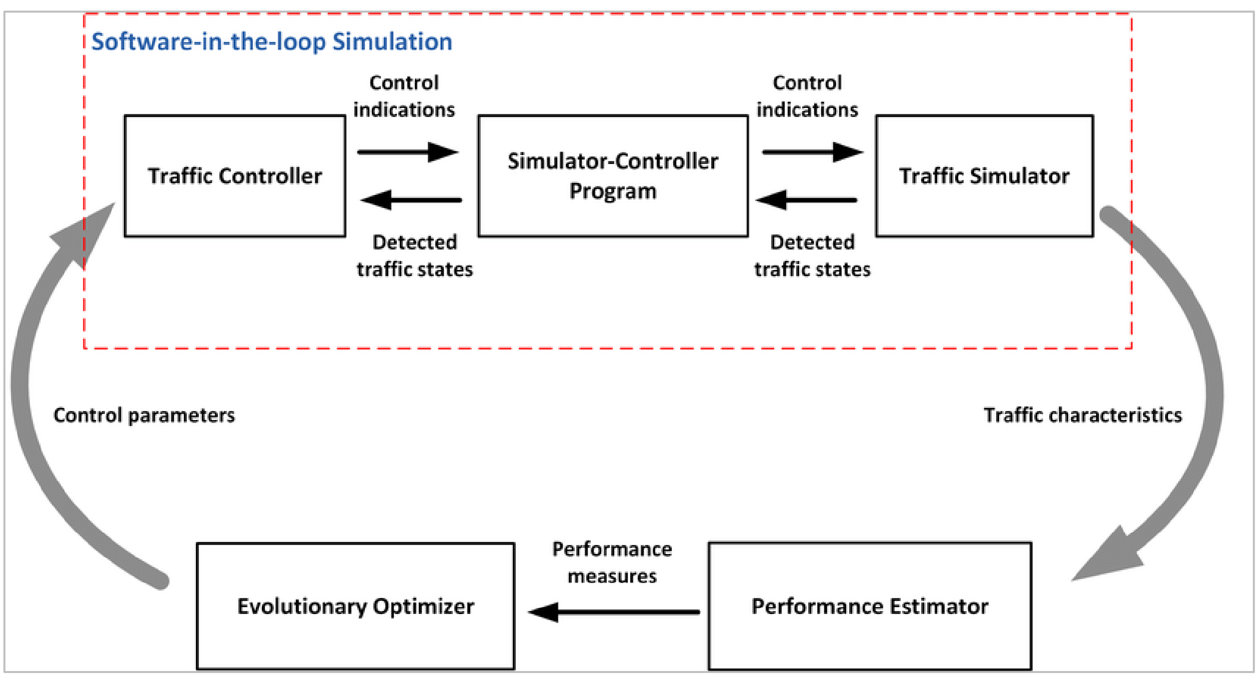


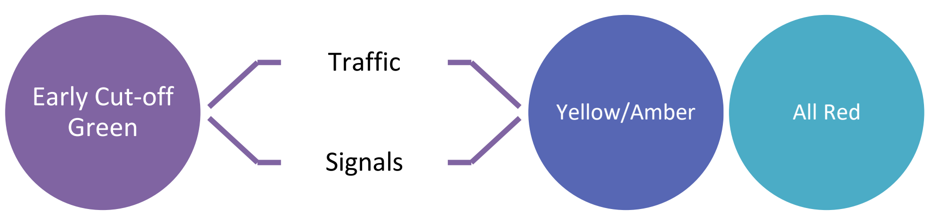
Figure 3: Software in Loop Simulation

1.1.2 Traffic Lights in Australia

Traffic lights were introduced in Australia on October 14, 1933. The first automatic traffic lights in Sydney were erected at the Central Business District's (CBD) Market Street and Kent Street crossroads. Although this crossroads was well recognised for its considerable traffic, it also posed special difficulties for horses travelling uphill from Pyrmont and Darling Harbour Roads because Market Street was paved with wooden blocks, providing a slick surface. On April 8, 1974, New South Wales (NSW) installed its one-thousandth traffic light. At NSW, there were 2,120 traffic signals at intersections as of August 1991 and an additional 280 were located at mid-block locations. In NSW, 3,791 traffic lights were operational as of December 2009. It took another four years before additional lights were put around the city and the nation, albeit the installation of traffic lights at the Sydney CBD crossroads intersection did improve traffic flow. Each potential direction of traffic flow is referred to as a "movement," and a phase is made up of a collection of conflict-free movements or a specific number of conflicting movements with clearly defined right-of-ways. The running part and the clearance part are the two fundamental components of each phase.

A phase's running portion consists of five consecutive time intervals:

The clearance part of the phase includes three sequential time intervals:



These intervals are essential for determining how traffic signals behave and guaranteeing secure and effective traffic control.

1.2Historical Context of Adding Amber Phase

Traditionally, traffic lights have featured three lenses: red, amber and green. However, limited research has explored traffic light optimisation with a "red-amber-green" sequence, likewise referred to as "amber before green." This strategy places an amber phase at signalised crossings after the red phase and before the green phase. This additional phase gives motorists a margin of time to respond and clear the intersection before the green light turns on. By using the amber light as a warning signal to prepare drivers, this idea aims to incorporate a variety into the minimum green and late start phases. This pattern will be followed by the new sequence:
Red -> Amber -> Green -> Yellow -> Red

Studies starting with amber phase studies on traffic signal systems have been carried out in several nations, including the UK, Germany and Belgium. The Department for Transport in the UK carried out a substantial amount of research in the 1960s. However, the original hypotheses based on this research are no longer applicable due to improvements in car technology, signal technology and driving experiences.

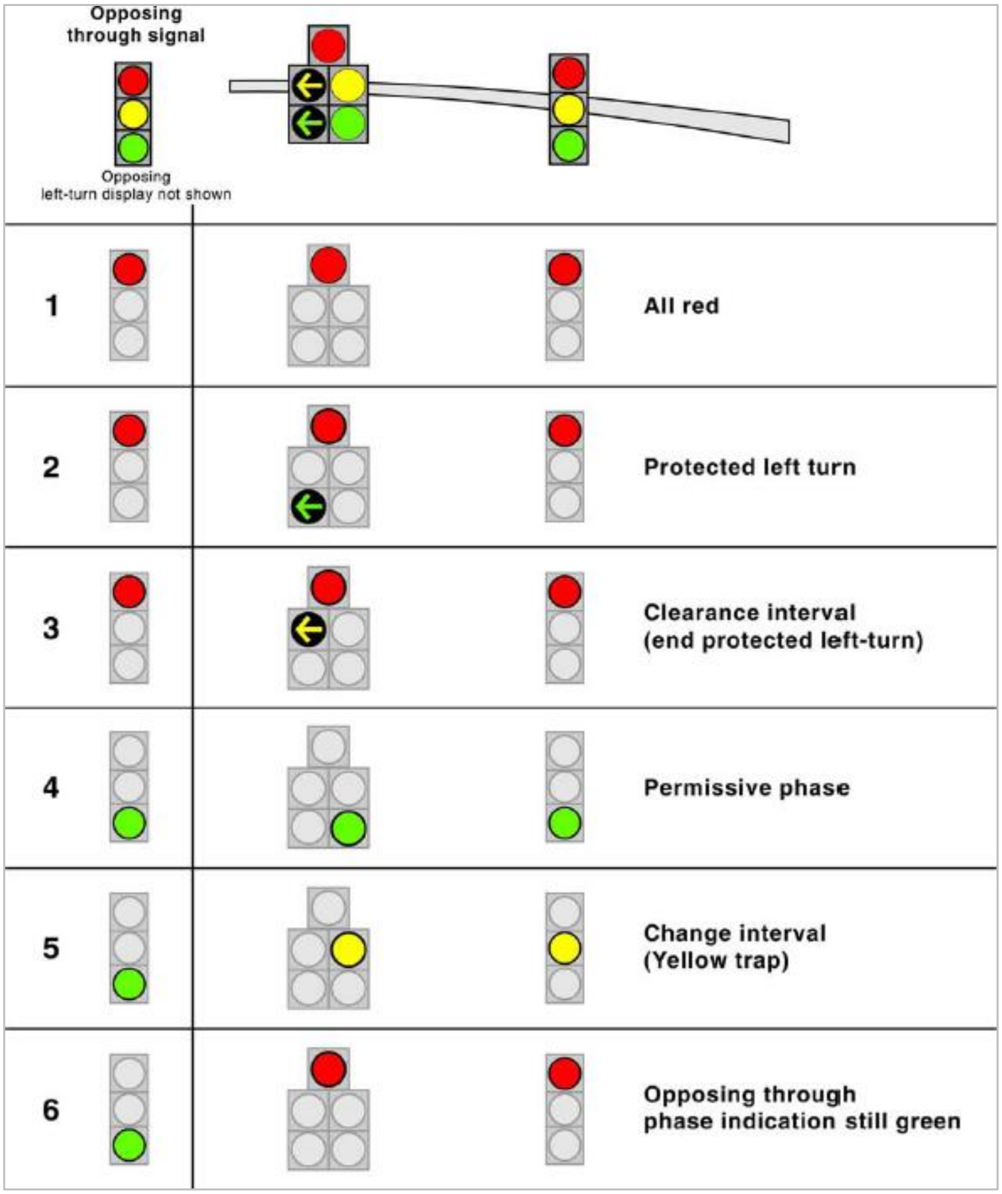


Figure 4: yellow trap Traffic Signal

The purpose of this study is to ascertain the effects on safety and effectiveness of reducing or changing the length of the initial amber phase. Road traffic lights are used by pan-European principles known as the "Vienna Convention," which was created by the European Conference of Ministers of Transport in 1974 (Wu, 2022). The commencing amber stage is discretionary and there are no required time intervals for its duration, per these regulations. When the red and amber phases are displayed together, it signals that a change in the signal is going to occur but does not remove the red light's ban on passing (Maxwell, 2005) (Agen, 1983).

1.2.1 Global Use of Amber Phase

The use of the amber phase varies worldwide, as illustrated in the figure below:

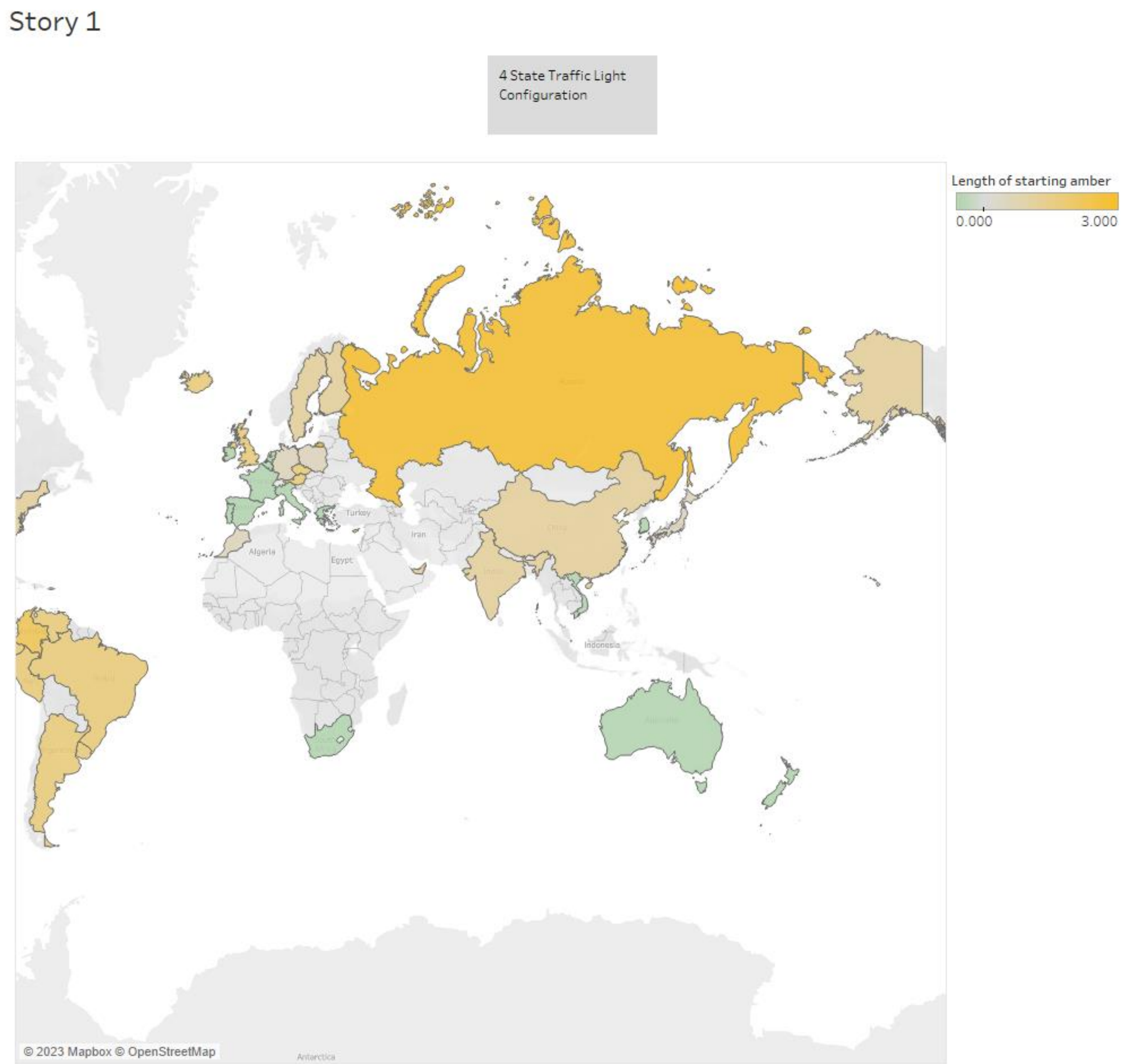


Figure 5 Worlwide yellow phase distriution. Own construction

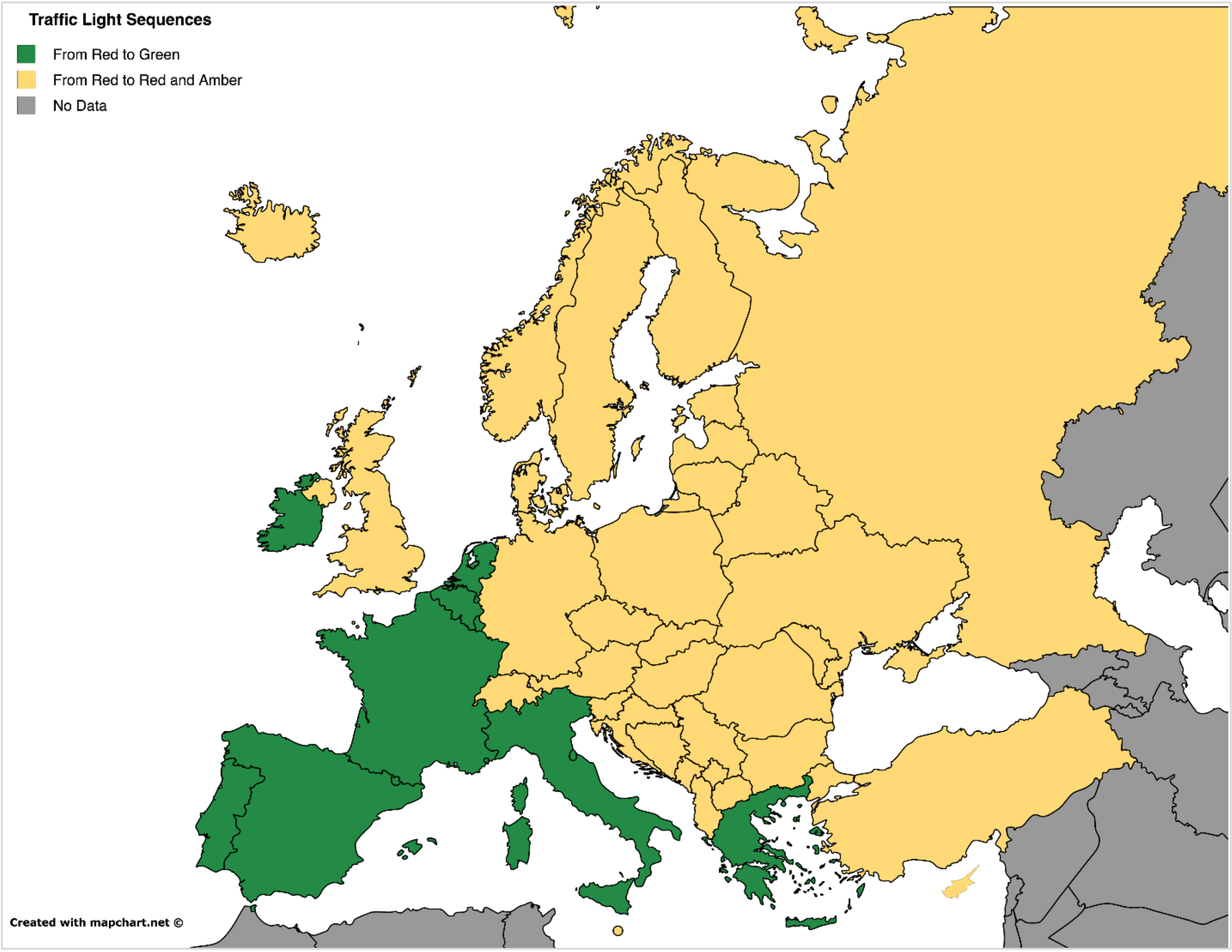


Figure 6: Countries Using the Red-Amber-Green Colour Scheme

Most European and South American countries include the traffic light cycle is red-amber. The initial amber is not used in Southern Europe, Canada, Australia, New Zealand, or South Africa. Numerous nations have conducted substantial research on this phase's consequences, such as the UK (1959-1962), Hong Kong (1974) and Melbourne (1970) . Research, including studies by Agent and Spelmans, highlights several advantages of including a starting amber phase. One significant benefit is the decrease of start-up lost time, which is the period lost as a result of a human response that is delayed when a traffic signal changes. According to Agent's research, adding the amber phase can reduce startup time by about 3 to 4 seconds. When the beginning amber was present, the start-up absences were reduced by about 1.1 seconds, according to Spelmans' driver simulation testing. The public opinion that drivers are better prepared and able to react quickly to the changing signal supports this beneficial effect on traffic flow. These results highlight the potential advantages of commencing traffic light systems with an amber phase. Drivers can anticipate the green light and have a quicker reaction time by being given a preliminary phase, which improves traffic flow. A one- or two-second all-red period has even been shown to reduce accidents between vehicles on separate highways by 83%, according to research. Additionally, different age groups of drivers may react to the starting amber arrangement differently, with younger drivers favouring it because it increases intersection capacity and lowers stress levels. The length of the amber gap varies on how fast the traffic is moving and in some nations, the red-yellow signal means that the green light will soon turn on. The red-yellow interval can be as little as 1.0 seconds or as long as 3.0 seconds in Russia. Traffic engineers found that the usual sequence successfully controls traffic without the requirement for an additional amber light before green after taking into account the traffic flow and congestion patterns in Australia (Maxwell, 2005) (Spelmans, 2017) (Luttinen, 2002).

2. CHAPTER III: METHODOLOGY

2.1Project Timeline

The project timeline spans approximately 9 months, with distinct deadlines allocated to each task. The initiation phase, encompassing the Literature Review, is estimated to take around two months. After that, it's predicted that the Data Collection and Analysis phase will take three months to complete. Next, it is anticipated that the Simulation Modelling phase will take around 2 months and that Hypothesis Testing will take another 2 months. Finally, the Conclusion and Recommendations segment is expected to be completed in 1-2 weeks. Throughout the project's lifecycle, regular meetings will be convened to review progress, identify potential challenges and ensure that the project remains on track

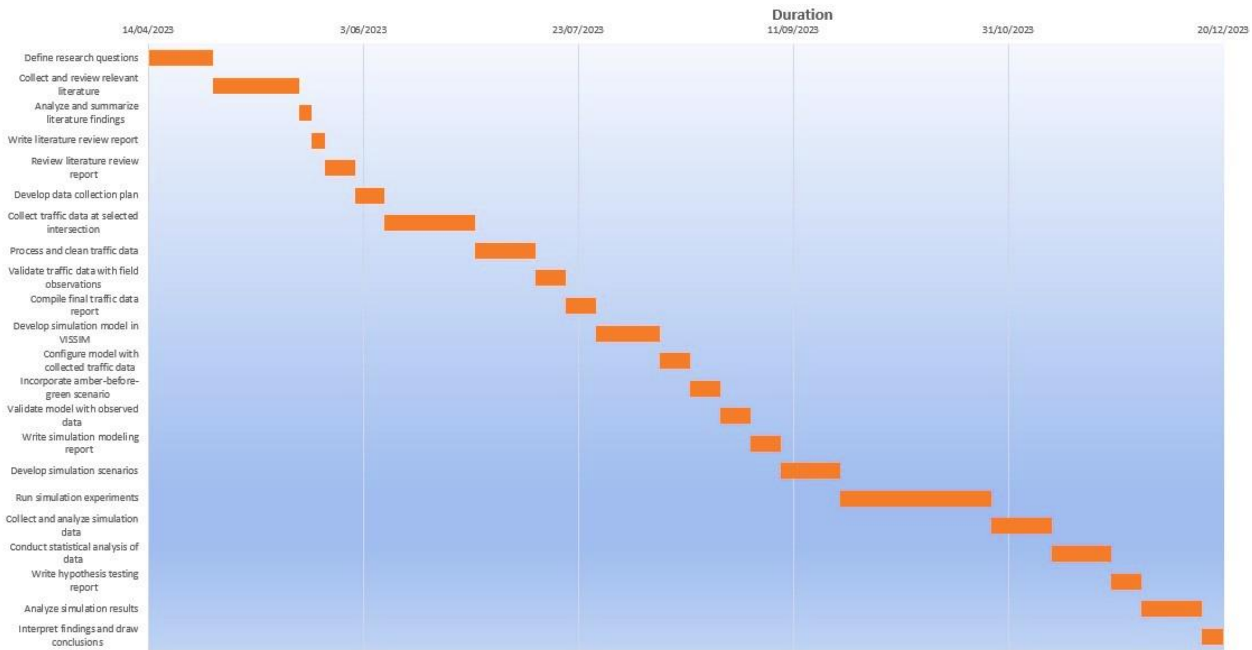


Figure 7 Figure 12: Gant Chart

2.2 Review of Existing Literature

In this research methodology, we initiate our study with an extensive literature review. Here, it is intended to develop an in-depth knowledge of traffic light optimisation techniques, simulation-based traffic flow analysis and the potential advantages of adding an amber light before the green phase. We use a methodical approach, carefully looking through *pertinent* databases and sources—both public and unpublished. Peer-reviewed journal papers, conference proceedings, technical reports and other reliable sources were all included in our evaluation. In the end, established traffic light optimisation approaches, simulation methodologies and insights about the introduction of an amber-before-green sequence have been identified through a critical analysis and synthesis of the findings. These findings will serve as the bedrock upon which we build our robust research methodology.

2.3Data Gathering

The study carried out an extensive data collection process to serve as the foundation for our inquiry. The main goal is to gather traffic information from a selected Sydney intersection to assess the impact of our research hypothesis. We employ appropriate data collection techniques, including traffic volume counting, traffic flow measurement and timing of traffic signal operations. These methods will guarantee that we amass an adequate dataset for simulating diverse traffic scenarios. Beyond the fundamental data, we also gather information about traffic behaviour, including vehicle speeds and types, enriching our understanding of traffic dynamics at the chosen junction.

Data Collection Procedure as Follows:

- We begin by identifying an appropriate junctions within Sydney for our data collection efforts.
- At the chosen intersections, we can install specialised traffic monitoring tools like loop detectors or traffic counters or just do it manually. These instruments will record important data.
- We meticulously record data on traffic volume, flow rates and signal timings. We ensure that data collection spans a sufficiently extensive period to encompass various traffic conditions and patterns.

This phase is expected to span several weeks to ensure the comprehensive gathering of data. Our commitment is to guarantee that the data collected is not only extensive but also maintains high standards of validity and reliability, aligning with the precise requirements of our research objectives.

3. CHAPTER IV: RESULTS

3.1 Data Analysis

The thorough investigation of the simulation findings is the focus of this phase. Descriptive statistics, t-tests and ANOVA were all employed to properly assess the efficacy of the suggested traffic signal optimisation strategy. In this currenbt study initially we collect data cleaning the simulation data and to do statistical analysis, the appropriate software or computer languages are used. In this report the usefulness of the improved signal timings while accounting for traffic flow rates, delays, queue wait times and other significant performance metrics. This report employ the appropriate statistical tests to compare the performance of improved signal timings to that of traditional signal timings. Regression analysis is used to identify key factors that influence the flow of traffic and its delay and hypothesis and study objectives, the analysis results are assessed. Descriptive statistics, t-tests, ANOVA and regression analysis were utilised to fully assess the effectiveness of the proposed traffic signal optimisation technique.

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
Vehicle Count	100	10	60	34.22	15.431
Pedestrian Count	100	10	25	17.61	4.524
Traffic Light Timing for Green Signal	100	20	60	42.79	14.385
Traffic Light Timing for Red Signal	100	10	40	24.17	8.300
Valid N (listwise)	100				

Figure 8 Summary Data collected

Table 1 Frecuency Road Intersection

Road Intersection

		Frequency	Per cent	Valid Percent	Cumulative Percent
Valid	Ilawarra Rd & Warren Rd	16	16.0	16.0	16.0
	Marrickville Rd & Livingston Rd	14	14.0	14.0	30.0
	Victoria Rd & Marrickville Rd	19	19.0	19.0	49.0
	Macquarie St & Smith St	26	26.0	26.0	75.0
	Hassal St & Smith St	25	25.0	25.0	100.0
	Total	100	100.0	100.0	

Table 2 Frecuency Weather

Weather

		Frequency	Per cent	Valid Percent	Cumulative Percent
Valid	Clear	23	23.0	23.0	23.0
	Cloudy	18	18.0	18.0	41.0
	Foggy	23	23.0	23.0	64.0
	Rainy	19	19.0	19.0	83.0
	Sunny	17	17.0	17.0	100.0
	Total	100	100.0	100.0	

Table 3 Fecuency Traffic Incidents

Traffic Incidents

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Minor Collision	6	6.0	6.0	6.0
	None	30	30.0	30.0	36.0
	Road Closure due to repair	64	64.0	64.0	100.0
	Total	100	100.0	100.0	

Table 4 Frecuency table Environmental Factors

Environmental Factors

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Clean Air	16	16.0	16.0	16.0
	High Humidity	13	13.0	13.0	29.0
	Low Visibility	21	21.0	21.0	50.0
	Mild Pollution	22	22.0	22.0	72.0
	Moderate Air Quality	13	13.0	13.0	85.0
	Moderate Pollution	15	15.0	15.0	100.0
	Total	100	100.0	100.0	

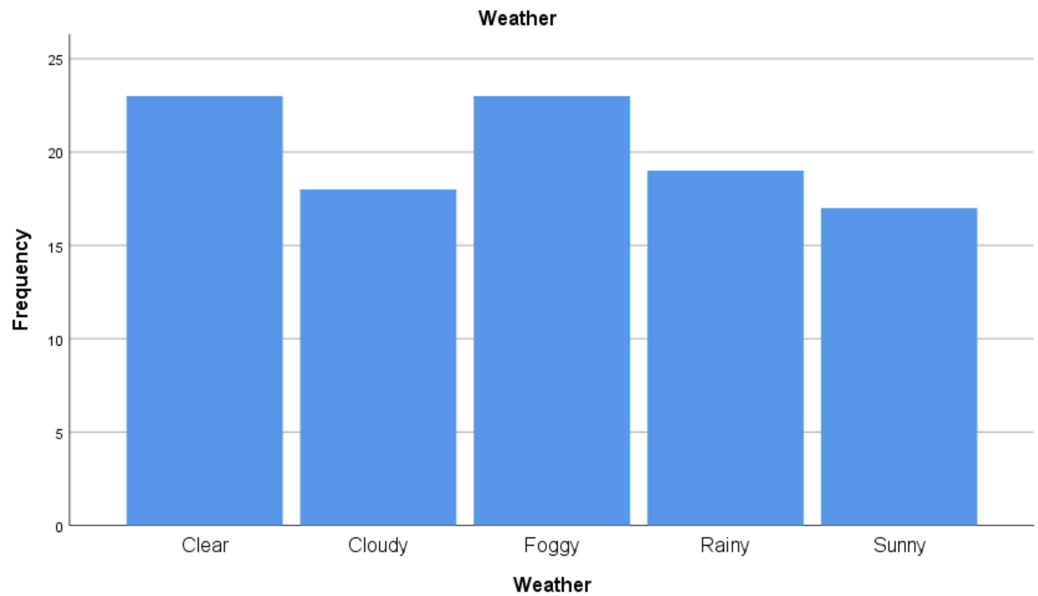


Figure 9: Weather Condition

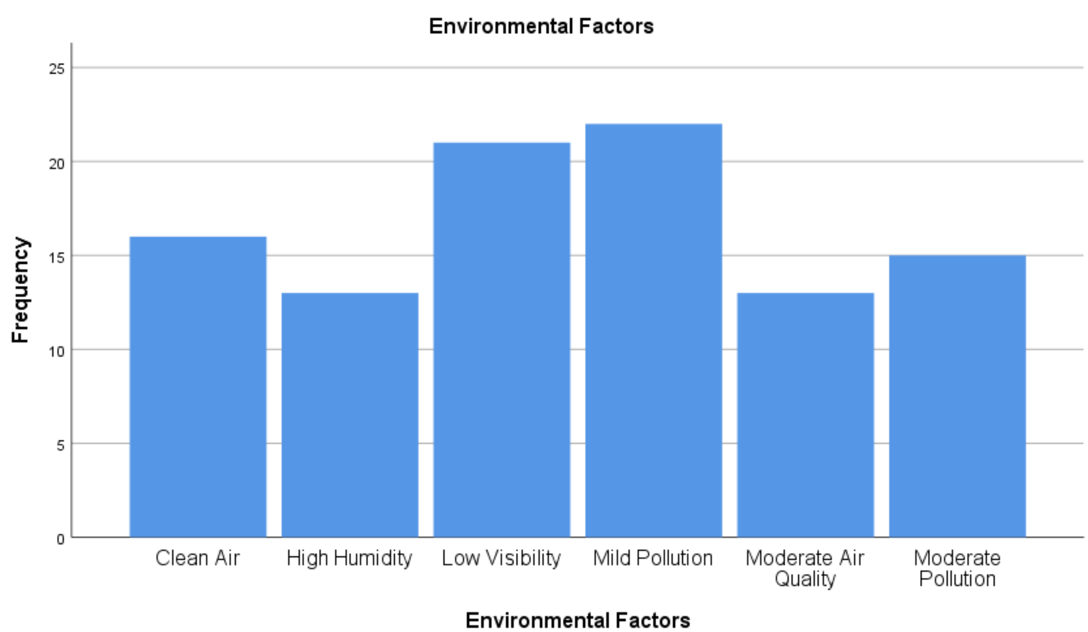


Figure 10: Environmental Factors

T-Test

As part of this research it will be important to consider if is an existing difference in Red light timing and Green light timing means, for this the author have conducted couple test using R showed below:

- Red Light Timing

```
        Welch Two Sample t-test

data:  TLTR by Suburb
t = -1.6397, df = 97.649, p-value = 0.9479
alternative hypothesis: true difference in means between group Marrickville and group Parramatta is greater than 0
95 percent confidence interval:
 -5.42284      Inf
sample estimates:
mean in group Marrickville   mean in group Parramatta
                  22.79592                   25.49020
```

- Green Light Timing

```
        Welch Two Sample t-test

data:  TLTG by Suburb
t = -1.292, df = 96.985, p-value = 0.9003
alternative hypothesis: true difference in means between group Marrickville and group Parramatta is greater than 0
95 percent confidence interval:
 -8.478601       Inf
sample estimates:
mean in group Marrickville   mean in group Parramatta
                  40.89796                   44.60784
```

As we can proved there is existing evidence that there is a difference between the means in the 2 suburbs.

- Box Plots

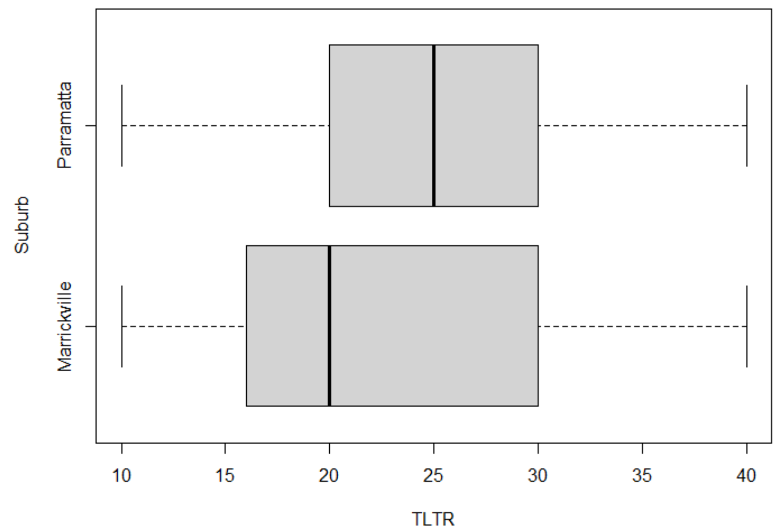


Figure 11 Suburb - Red Timing

As we can see in the previous graph mostly of the observation are located above the median indicating the skewness of our sample and the not normal distribution. In the Graph below we can see the data is not normal distributed.

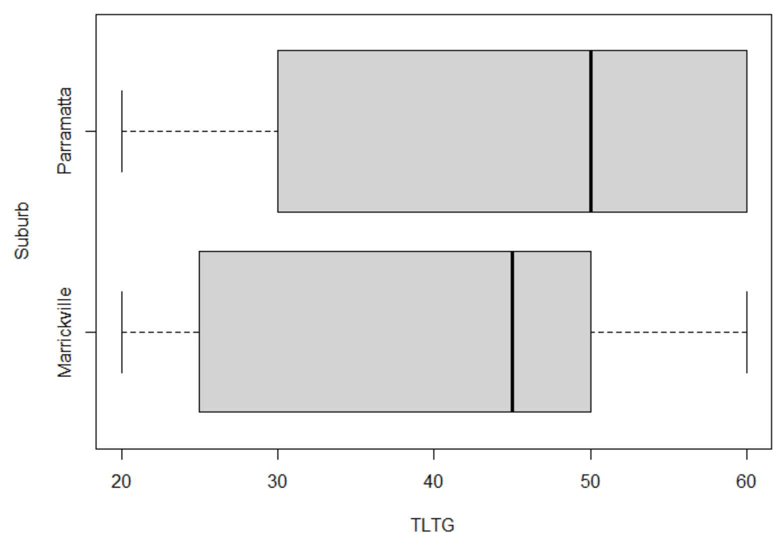


Figure 12 Suburb - Green Timing

In order to see if the weather influence the behaviour of the lights and there is and existing change if the weather is not the same the author have conducted a comparison using the ANOVA test:

Oneway

Table 5 Oneway Anova Analysis

		N	Mean	Std. Deviation	Std. Error
Traffic Light Timing for Red Signal	Clean Air	16	20.88	5.749	1.437
	High Humidity	13	26.62	9.386	2.603
	Low Visibility	21	25.81	10.078	2.199
	Mild Pollution	22	24.59	7.639	1.629
	Moderate Air Quality	13	24.46	9.422	2.613
	Moderate Pollution	15	22.40	6.468	1.670
	Total	100	24.17	8.300	.830
Traffic Light Timing for Green Signal	Clean Air	16	38.63	12.500	3.125
	High Humidity	13	43.62	15.804	4.383
	Low Visibility	21	42.10	15.643	3.413

	Mild Pollution	22	45.82	13.972	2.979
	Moderate Air Quality	13	44.23	15.922	4.416
	Moderate Pollution	15	41.80	13.550	3.499
	Total	100	42.79	14.385	1.439

		Sum of Squares	df	Mean Square	F
Traffic Light Timing for Red Signal	Between Groups	359.896	5	71.979	1.047
	Within Groups	6460.214	94	68.726	
	Total	6820.110	99		
Traffic Light Timing for Green Signal	Between Groups	539.973	5	107.995	.509
	Within Groups	19946.617	94	212.198	
	Total	20486.590	99		

3.2 Simulation Model

In this phase, we develop an scenario using use the proper simulation tools or programming languages like Pyhton, using the library PyGame to implement the stochastic traffic simulation model, a microsimulation model. Microsimulation models are well-suited for studying individual vehicle movements and interactions at a detailed level, making them particularly useful for traffic optimization studies.

This tool have been developed by Mihir Gandhi, Solanki Devansh, Daptardar Rutwij and Nimala Baloorkar (Gandhi, et al., 2020), their research in Traffic Engineering, result in a very handy tool that allowed the visualization and simulate from scratch using Pygame to display the movement of vehicles at a traffic intersection. The intersection has traffic lights with timers controlling the traffic flow in all four directions. Each signal displays the time remaining for it to change from green to yellow, yellow to red, or red to green or in our scenario red to yellow and yellow to green. Various vehicles like cars, bikes, buses, and trucks are generated, and their movement is influenced by the signals and the presence of other vehicles. This simulation have been be utilized for data analysis or to visualize in terms of this project use.

Pygame is a versatile collection of Python modules tailored for game development, offering computer graphics and sound libraries specifically compatible with the Python programming language. Pygame extends the capabilities of the SDL library, enabling users to develop comprehensive games and multimedia applications using Python. Its portability is a key feature, allowing it to function on various platforms and operating systems. Pygame is both free to use and licensed under LGPL (Pygame Developers, 2019)

Simulation Model Development Procedure:

In the development of our simulation model, we represent a sample Sydney traffic junctions, mirroring its existing characteristics. To achieve this, we meticulously collected and processed real-world traffic data encompassing road network layouts, vehicle counts, and current traffic signal timings. Before introducing the whole tool and the code is clear to explain the tool, and display some of the images needed for the construction of our scenario:

- Normal and Simple Intersection

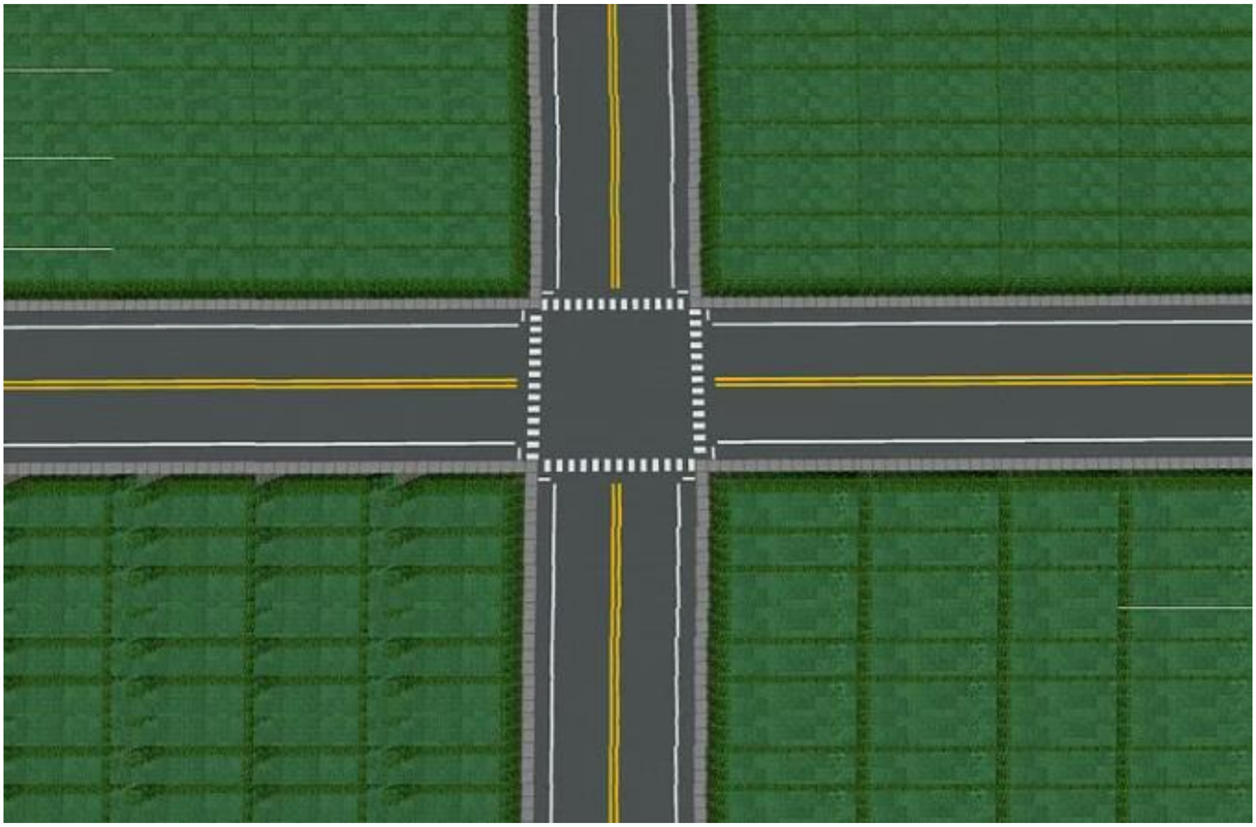


Figure 13 Simple Intersection

- Traffice Signals: Red, Yellow and Green

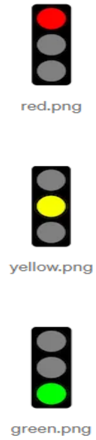


Figure 14 Traffic Light Phases

- Vehicles

Table 6 Vehicles in the Model Simulation

CAR	
BIKE	
BUS	
TRUCK	

The provided tool is a Python script is displayed as part of the APPENDIX 1 for simulating traffic flow at an intersection. Breaking it down the key components and functionalities of this tool is explained below:

1. Initialization:

The script initializes various parameters related to the traffic simulation, such as signal timers, vehicle speeds, lane configurations, and graphical elements.

2. Traffic Signal Logic:

- **TrafficSignal Class:** Represents the traffic signals with attributes for red, yellow, and green timers. These timers dictate how long each signal state lasts.

- **Signal Switching Logic**: The `repeat()` function controls the switching of traffic signals. It decrements the timers for green, yellow, and red signals, updating the signal state based on the timers.

3. Vehicle Generation and Movement:

- **Vehicle Class:** Represents a vehicle with attributes like lane, vehicle class (car, bus, etc.), speed, direction, and turn indicator. Vehicles are generated randomly based on certain probabilities and move within their lanes.

- Vehicle Movement: The `move()` method of the Vehicle class determines how vehicles move within their lanes and stop at red signals.

4. Simulation Display:

- **Pygame:** The simulation is displayed using the Pygame library. It sets up the simulation window, loads images for signals and vehicles, and continuously updates the screen to visualize the traffic flow.

5. Multithreading:

The simulation utilizes multithreading to handle different tasks concurrently:

- **Thread 1 (`initialization`):** Initializes the traffic signals and starts the signal switching logic.

- **Thread 2 (`generateVehicles`):** Generates vehicles randomly based on probabilities.

- **Thread 3 (`simTime`)**: Keeps track of the simulation time and exits the program after a predefined simulation duration.

6. User-Defined Parameters:

- **Signal Timers:** You can set default green, yellow, and red signal timers as well as a random range for green signal timers.

- Allowed Vehicle Types: Specifies which types of vehicles (car, bus, truck, bike) are allowed in the simulation.

- **Intersection Layout:** Defines the coordinates and lanes for vehicles, signals, and stop lines.

7. Simulation Execution:

The `Main()` class initializes the simulation and starts the threads for different tasks. The Pygame window continuously updates to reflect the current state of the simulation.

In summary, the script creates a traffic simulation at an intersection, where vehicles move, stop at signals, and follow predefined rules based on signal timers and lane configurations. The simulation continues until a specified duration, displaying real-time traffic flow and signal states. This simulation was created as a component of a research endeavor titled 'Intelligent Traffic Signal Control Using Artificial Intelligence' (Gandhi, et al., 2020). This research initiative was featured at the IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE) in 2020 and subsequently published on IEEE Xplore.

There are some modifications done by the author according to the project objective, some of them are explained as follows

Differences in the Amber before Green Code:

Table 7 Amber - Green Code Explication

Item	**Previous Code**	**Updated Code**
Modularization and Classes	The code was written in a single script without any modularization or classes.	The code has been organized into classes (`TrafficSignal` and `Vehicle`). This object-oriented approach enhances readability, reusability, and maintainability of the code.
• **Multithreading:**	Multithreading was not implemented in the previous code.	Multithreading is utilized in the updated code. Threads are used for functions like initialization, vehicle generation, and simulation time, enabling concurrent execution of these tasks.
• **Simulation Logic:**	The logic for traffic signal simulation and vehicle movement was implemented in a procedural manner, making it complex to manage	The simulation logic is encapsulated within the `TrafficSignal` and `Vehicle` classes. This encapsulation improves code organization and readability.
Signal Logic Enhancement	The signal logic was rudimentary and lacked features like transitioning	The updated code includes logic for transitioning between green, yellow, and red signals based on specified

	between signal states (green, yellow, red).	timers. The simulation runs through signal states (green, yellow, red) in a controlled manner, enhancing realism
Vehicle Generation	Vehicle generation was handled using randomization without a clear structure.	Vehicles are generated using a more structured approach. Vehicle types, lanes, and directions are chosen randomly, but within specific ranges, creating a more controlled and realistic vehicle flow.
Simulation Time Limit:	There was no specified time limit for the simulation.	The simulation now has a specified time limit (`simulationTime` variable) after which the simulation ends and displays statistics
Lane Counters:	Lane counters were not implemented in the previous code.	Lane counters are implemented and displayed on the simulation screen, showing the number of vehicles that have crossed each lane
Code Organization:	The code lacked clear organization and structure, making it difficult to follow the flow of execution.	The updated code is organized into functions and classes, providing a clear structure and making it easier to comprehend and modify

		specific parts of the simulation
Enhanced Vehicle Movement:	Vehicle movement logic was limited, and there were gaps between vehicles without proper stopping and moving distances.	The updated code includes enhanced vehicle movement logic, ensuring proper stopping and moving distances between vehicles, improving the realism of traffic flow.
Simulation Termination	The simulation ran indefinitely without a defined end point.	The simulation in the updated code has a defined time limit (`simulationTime`), after which it displays statistics and terminates.

- **New Traffic Signal Phase: Yellow Phase**

In the updated simulation code, a new phase, known as the Yellow Phase, has been introduced between the red and green signal phases. This addition enhances the simulation's realism and mimics real-world traffic signal systems that we are aiming for. During the yellow phase, the yellow signal is displayed, indicating an impending signal change. This phase serves as a transition period, allowing vehicles to slow down and come to a stop before the signal changes from red to green or vice versa. This enhancement **promotes** safer and more efficient traffic flow at intersections, aligning the simulation more closely with real-world traffic management practices.
These enhancements demonstrate a significant improvement in code organization, functionality, and realism in the updated version of the simulation. The use of classes, multithreading, and graphical representation enhances the overall quality and user experience of the simulation

Integration of Metrics into the Traffic Control System:

In the enhanced traffic simulation model, the metrics of Idle Time, Perception-Reaction Time, and Start-up Lost Time have been seamlessly integrated to create a more responsive and efficient traffic control system. Here's how these metrics have been incorporated into the new scenario:

1. Idle Time Optimization:

- Implementation: By continuously monitoring the movement of vehicles at the intersection, the system minimizes idle time. Vehicles are allowed to move as soon as the signal changes, reducing the waiting period significantly.
- Impact: Reduced idle time ensures vehicles spend minimal time stationary, promoting a continuous flow of traffic. This optimization is achieved by dynamically adjusting signal timings based on real-time traffic conditions.

2. Perception-Reaction Time Consideration:

- Implementation: The traffic control system accounts for the perception-reaction time of drivers when signals change, especially during the transition from red to green. Signals are timed to allow for this reaction time, enabling vehicles to initiate movement promptly after the signal switches.
- Impact: By synchronizing signal changes with the drivers' perception-reaction time, the system minimizes the delay between the signal shift and the commencement of vehicle movement. This synchronization reduces reaction-related delays, enhancing the overall efficiency of the intersection.

3. Start-up Lost Time Reduction:

- Implementation: Start-up lost time, representing the delay between signal change and the first vehicle crossing the stop line, is meticulously minimized. The system ensures swift acceleration of the first vehicle, allowing it to cross the stop line promptly after the green signal activates.
- Impact: By reducing start-up lost time, the traffic control system mitigates the delay experienced by the first vehicle, enabling it to move swiftly from a complete stop. This reduction optimizes the intersection's efficiency by minimizing initial acceleration delays.

Overall System Adaptability:

- Dynamic Adjustments: The traffic control system continuously analyzes the traffic dynamics, adapting signal timings based on the integrated metrics. Real-time adjustments ensure that the system responds promptly to changing traffic patterns, maximizing intersection throughput and minimizing delays.
- Safety Enhancement: By incorporating perception-reaction time and minimizing start-up lost time, the system enhances safety. Swift and predictable vehicle movements reduce the likelihood of abrupt stops, minimizing the risk of collisions and ensuring a safer traffic environment.

Below is display the simulation output after modified all this parameter in order to create the needed scenario:

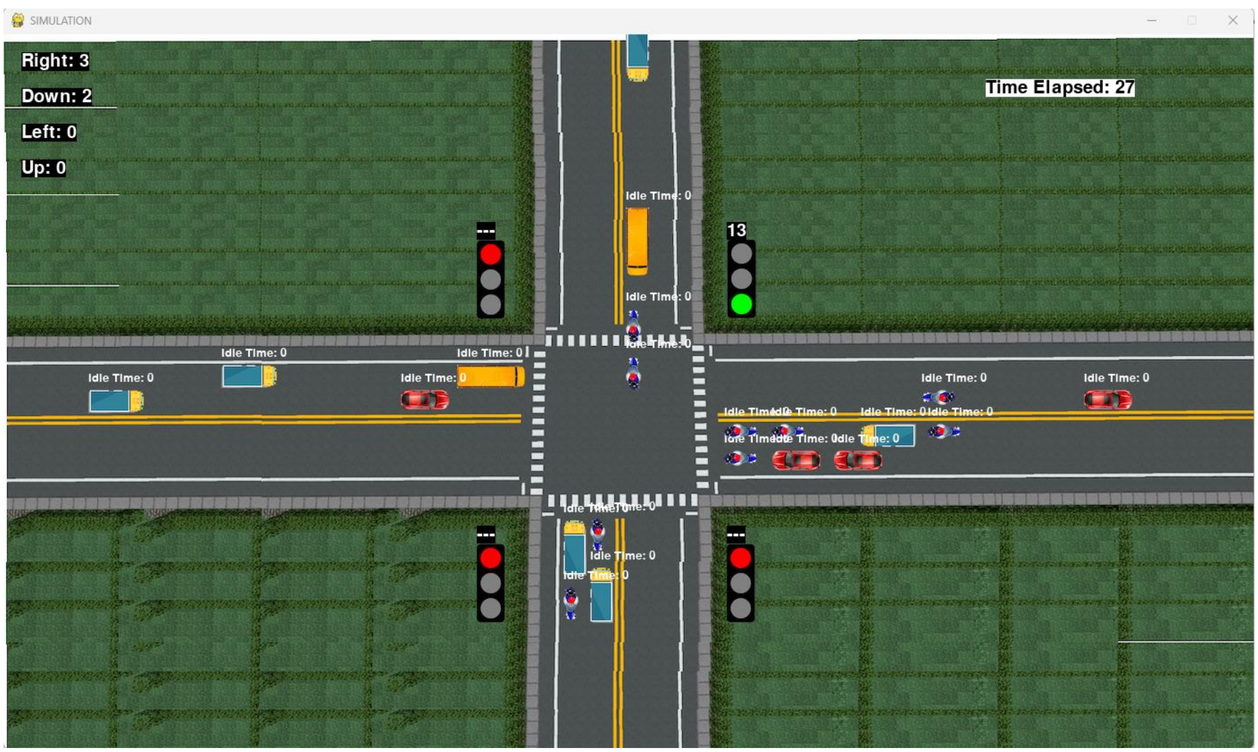


Figure 15 Simulation Output

3.3 Evaluation of the proposed scenario

To evaluate the effectiveness of the scenario Amber – Green compared to the Red – Green scenario, this author have picked 15 simulations for both scenarios, each lasting 5 minutes. During these simulations, traffic patterns were generated randomly. The performance assessment focused on the number of vehicles that successfully crossed the intersection within a specific timeframe. Specifically, is analyzed the the start up lost time and the idle time of the traffic signal, which refers to the duration when the signal is green but no cars pass the intersection.

Before analyzing the data obtained with this simulation is important to analyze and understand how vehicles are distributed across lane at the intersection and junction, is crucial for modeling the close real world scenarios. The distribution, denoted as [a,b,c,d] represents the probability of vehicles being at an specific lanes, with probability a/d, (b-a)/d, (c-b)/d and (d-c)/d for each lane in the intersection. These probabilities reflect the expected value and the likelihood of finding a vehicle in each lane, offering insights into realistic traffic patterns. By considering this distribution, researchers and engineers can calculate the expected value representing the number of vehicles in each lane. This value, derived by multiplying each possible lane occupancy by its respective probability and summing these values, serves as a central measure in the distribution. The expected value is computed using the formula: $E(X)=a/d\times1+(b-a)/d\times2+(c-b)/d\times3+(d-c)/d\times4$.
This measure serves as a central tendency in the distribution, enabling experts to optimize intersection designs, signal timings, and traffic management strategies for more efficient and realistic outcomes. (Gandhi, et al., 2020) (kumar & Tejaswini, 2023).

Table 8 SIMULATION TABLE 2. SCENARIO RED – GREEN TRAFFIC LIGHT OPTIMIZATION

Simulation	Distribution	Lane 1	Lane 2	Lane 3	Lane 4	Total
1	[25,45,65,100]	10	15	20	26	71
2	[35,55,75,100]	14	18	23	26	81
3	[20,40,70,100]	8	12	19	25	64
4	[30,50,80,100]	12	15	25	28	80
5	[15,35,65,100]	6	11	18	27	62

6	[40,60,80,100]	16	19	24	21	80
7	[22,42,72,100]	9	14	21	26	70
8	[28,48,78,100]	11	15	24	26	76
9	[18,38,68,100]	7	11	18	24	60
10	[32,52,82,100]	13	17	26	24	80
11	[23,43,73,100]	9	12	20	28	69
12	[38,58,78,100]	15	18	24	23	80
13	[26,46,76,100]	10	13	21	28	72
14	[29,49,79,100]	12	16	22	25	75
15	[24,44,74,100]	9	14	21	28	72

Table 9 SIMULATION TABLE 1. SCENARIO AMBER – GREEN TRAFFIC LIGHT OPTIMIZATION

Simulation	Distribution	Lane 1	Lane 2	Lane 3	Lane 4	Total Vehicles
1	[25,45,65,100]	13	16	21	26	76
2	[35,55,75,100]	17	18	23	26	84
3	[20,40,70,100]	15	14	20	25	74
4	[30,50,80,100]	15	17	26	31	89
5	[15,35,65,100]	13	14	21	32	80
6	[40,60,80,100]	18	20	25	23	86
7	[22,42,72,100]	14	16	23	27	80
8	[28,48,78,100]	14	18	27	29	88
9	[18,38,68,100]	15	15	22	28	80

10	[32,52,82,100]	17	19	28	26	90
11	[23,43,73,100]	15	16	24	29	84
12	[38,58,78,100]	20	18	24	23	85
13	[26,46,76,100]	16	14	22	28	80
14	[29,49,79,100]	17	17	23	27	84
15	[24,44,74,100]	14	16	23	30	83

As we can see in the table above, the Amber - Green scenario performs slightly better than the current system once is include the starting up lost time of 4.5 segs and a yellow phase less than 2.5 . the improvement in performance can be seen also in the skewness of the traffic distribution.

1. **Traffic Volume and Distribution:**

- The analysis of Table 1 and Table 3 reveals distinct patterns in traffic volume and distribution. Table 3 consistently accommodates a higher volume of vehicles across various scenarios, while Table 1 maintains a more balanced and stable distribution of vehicles across lanes.

2. **Stability vs. Variability:**

- Table 1 demonstrates a more stable and predictable traffic flow due to its narrower distribution ranges. In contrast, Table 3, with its wider ranges, handles higher traffic volumes but exhibits increased variability and unpredictability in traffic patterns.

3. **Efficiency and Predictability:**

- Table 1 is suitable for scenarios where traffic stability and predictability are paramount. The controlled distribution of vehicles ensures a smoother traffic flow, reducing congestion and enhancing overall efficiency. Table 3, while capable of handling higher traffic volumes, may experience variable traffic patterns, impacting predictability.

4. **Tailoring Strategies to Goals:**

- The choice between Table 1 and Table 3 should align with specific objectives. If the priority is on maintaining a stable and balanced traffic flow, Table 1 offers an efficient solution. For situations requiring the accommodation of higher traffic volumes at the cost of some variability, Table 3 could be the preferred choice.

5. **Continuous Monitoring and Adaptation:**

- Regardless of the chosen approach, continuous monitoring of traffic data is essential. Regular analysis and adaptation of distribution strategies based on real-time traffic patterns will enable the traffic management system to remain responsive to changing demands and ensure optimal performance.

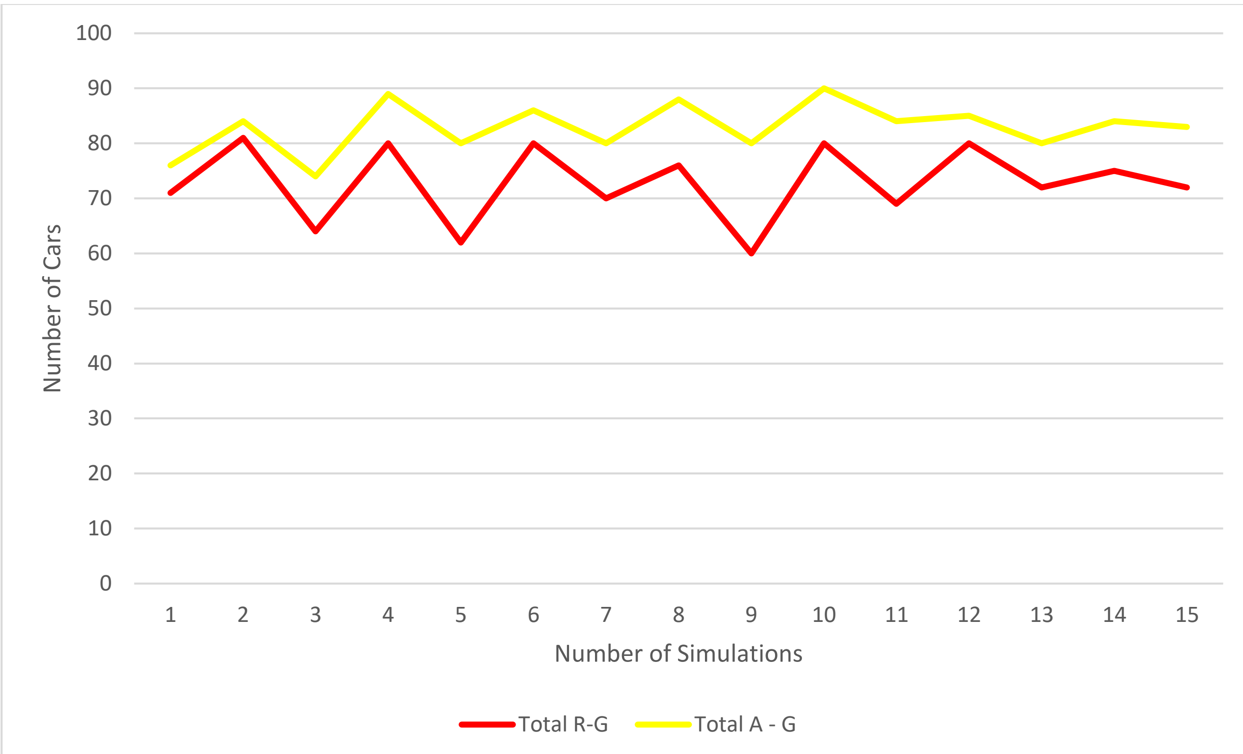


Figure 16 Simulation Camparison

Under the inclusion of the different features to each scenario in the simulation conditions like start up time, idle time and similar encompassing factors such as traffic distribution, vehicle speeds and inter-vehicle gaps, the simulations were conducted over a total duration of 1 hour and 15 minutes.

Each distribution scenario spanned 300 seconds, equating to 5 minutes. The results indicated that the proposed scenario, on average, demonstrated a performance enhancement of approximately 13% compared to the existing fixed-time system. This enhancement suggests a reduction in idle green signal time and a decrease in waiting time for vehicles at intersections.

CHAPTER V: CONCLUSION

Consistent with findings in existing literature, this study observed a positive influence of implementing an amber phase on traffic flow. When Drivers are improved readiness to respond to the upcoming green phase, it leads to a reduction in start-up lost time and less time wasted in the intersection. This decrease in start-up lost time was primarily attributed to drivers initiating their movement earlier, reducing also the idle time in this scenario.

This simulation study on the starting amber configuration has several constraints and assumptions were encountered. A practical on-site test was unfeasible due to the absence of a legal framework and resources. As a future work Is recommended to employ a driving simulator as a viable alternative, participants in driving simulator experiments or empirical studies that not tend to adopt a socially acceptable driving behaviour. Simulated environment led to disparities in participants' perception of speed and distance, differing significantly from real-world scenarios.

To conclude, the proposed scenario presents a different approach to traffic light phases, tailoring the phases and start up time longer than usual by definition. Prioritizing directions with higher traffic volumes, it slighlty optimizes signal time, minimizing delays, congestion, and waiting periods. This strategic approach not only enhances traffic flow efficiency but also contributes significantly to reduced fuel consumption and environmental pollution, marking a positive stride toward sustainable urban mobility. (Bouktif, 2023) (Ding, 2022) (Ducrocq, 2023) (Hong, 2022)

References

Bouktif, S. C. A. a. E.-S., 2023. Deep reinforcement learning for traffic signal control with consistent state and reward design approach. *Knowledge Based Systems* .

Ding, C. D. F. Z. a. W. X., 2022. Collaborative control of traffic signal and variable guiding lane for isolated intersection under connected and automated vehicle environment. *Computer - aided civil and Infrastructure Engineering* , Volumen 37, pp. 2052 - 2069.

Ducrocq, R. a. F. N., 2023. Deep reinforcement Q - learning for intelligent traffic signal control with partial detection. *International Journal of Intelligent Transportation Systems Research ,* Volumen 1, pp. 192 - 206.

Gandhi, M., Solanki, D., Daptardar, R. & Baloorkar, N., 2020. Smart Control of Traffic Light Using Artificial Intelligence. *IEEE International Conference on Recent Advances and Innovations in Engineering.*

Hong, W. T. W., 2022. Traffic signal control with addaptative online learning scheme using multiple model neural networks. *IEEE transactions on neural networks and learning systems.*

Kolat, M. K. B. B. T. a. A. S., 2023. Multi agent reinforcement learning for traffic signal control: A cooperative approach. *Sustainability,* pp. 34 - 79.

Korecki, M. D. D. a. H. D., 2023. How well do reinforcement learning approaches cope with disruptions? The case of traffic signal control. *IEEE Access.*

kumar, A. & Tejaswini, N. M., 2023. Smart Control of Traffic Light Using Deep Learning. *Industrial Engineering Journal* , pp. 59 - 63.

Lin, H. H. C. W. a. J. B., 2022. Traffic signal optimization based on fuzzy control and differential evolution algorithm. *IEEE.*

Liu, M. Z. H. S. a. W., 2022. An optimal control approach of integrating traffic signals and cooperative vehicle trajectories at intersections. *Transportmetrica B: transport dynamics,* pp. 971 - 987.

Pygame Developers, 2019. *https://www.pygame.org/wiki/about.* [En línea].

Qadri, S. G. M., 2020. State-of-art review of traffic signal control methods: Challenges and opportunities. *European transport research review* , Volumen 12, pp. 1 - 23 .

Wu, Q. W. J. S. J. D. B. T. A. F. M. a. L. C., 2022. Distributed agent-based deep reinforcement learning for large scale traffic signal control.. *Knowledge Based Systems* , Volumen 241, pp. 108 - 304.

Yang, S., 2023. Hierarchical graph multi agent reinforcement learning for traffic signal control. *Informational sciences,* pp. 55 -72 .

Yazdani, M. S. M. B. S. N. N. P. J. P. H., 2023. Intelligent vehicle pedestrian light (IVPL): A deep reinforcement learning approach for traffic signal control.. *Transportation research part C: emerging technologies* , pp. 103 - 991.

Ying, Z. C. L. M. M. J. a. D. R., 2022. . PrivacySignal: Privacy-preserving traffic signal control for intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems* , Volumen 23, pp. 16290 - 16303.

Zheng, L. a. L. X., 2023. Simulation-based optimization method for arterial signal control considering traffic safety and efficiency under uncertainties. *Computer Aided Civil and Infrastructure Engineering* , pp. 640 - 659.

Appendix A: CODE

.

In [1]:

```
import random
import time
import threading
import pygame
import sys
import os


lane_counters = {'right': 0, 'down': 0, 'left': 0, 'up': 0}


# Default values of signal timers
defaultGreen = {0:15, 1:15, 2:15, 3:15}
defaultRed = 150
defaultYellow = 5
PRT = 0  # Example PRT value (in seconds)

signals = []
noOfSignals = 4
currentGreen = 0   # Indicates which signal is green currently
nextGreen = (currentGreen+1)%noOfSignals    # Indicates which signal will turn green next
currentYellow = 0   # Indicates whether yellow signal is on or off

speeds = {'car':2.25, 'bus':1.8, 'truck':1.8, 'bike':2.5}  # average speeds of vehicles

# Coordinates of vehicles' start
x = {'right':[0,0,0], 'down':[755,727,697], 'left':[1400,1400,1400], 'up':[602,627,657]}
y = {'right':[348,370,398], 'down':[0,0,0], 'left':[498,466,436], 'up':[800,800,800]}

vehicles = {'right': {0:[], 1:[], 2:[], 'crossed':0}, 'down': {0:[], 1:[], 2:[], 'crossed':0}, 'left': {0:[], 1:[], 2:[], 'cr
vehicleTypes = {0:'car', 1:'bus', 2:'truck', 3:'bike'}
directionNumbers = {0:'right', 1:'down', 2:'left', 3:'up'}

# Coordinates of signal image, timer, and vehicle count
signalCoods = [(530,230),(810,230),(810,570),(530,570)]
signalTimerCoods = [(530,210),(810,210),(810,550),(530,550)]

# Coordinates of stop lines
stopLines = {'right': 590, 'down': 330, 'left': 800, 'up': 535}
defaultStop = {'right': 580, 'down': 320, 'left': 810, 'up': 545}
# stops = {'right': [580,580,580], 'down': [320,320,320], 'left': [810,810,810], 'up': [545,545,545]}
```

```python
# Gap between vehicles
stoppingGap = 15    # stopping gap
movingGap = 15   # moving gap

timeElapsed = 0
simulationTime = 300
timeElapsedCoods = (1100,50)

pygame.init()
simulation = pygame.sprite.Group()

class TrafficSignal:
    def __init__(self, red, yellow, green):
        self.red = red
        self.yellow = yellow
        self.green = green
        self.signalText = ""

class Vehicle(pygame.sprite.Sprite):
    def __init__(self, lane, vehicleClass, direction_number, direction):
        pygame.sprite.Sprite.__init__(self)
        self.lane = lane
        self.vehicleClass = vehicleClass
        self.speed = speeds[vehicleClass]
        self.direction_number = direction_number
        self.direction = direction
        self.x = x[direction][lane]
        self.y = y[direction][lane]
        self.crossed = 0
        self.idle_time = 0
        self.start_up_time = 5
        #self.control_delay = 0
        self.prt_countdown = PRT
        vehicles[direction][lane].append(self)
        self.index = len(vehicles[direction][lane]) - 1
        path = "images/" + direction + "/" + vehicleClass + ".png"
        self.image = pygame.image.load(path)

        if(len(vehicles[direction][lane])>1 and vehicles[direction][lane][self.index-1].crossed==0):    # if more than 1 vehi
            if(direction=='right'):
                self.stop = vehicles[direction][lane][self.index-1].stop - vehicles[direction][lane][self.index-1].image.get_
            elif(direction=='left'):
                self.stop = vehicles[direction][lane][self.index-1].stop + vehicles[direction][lane][self.index-1].image.get_
```

jupyter Amber_Green Model_01_11_2023-Copy1 Last Checkpoint: Yesterday at 5:34 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

```python
        # Set new starting and stopping coordinate
        if(direction=='right'):
            temp = self.image.get_rect().width + stoppingGap
            x[direction][lane] -= temp
        elif(direction=='left'):
            temp = self.image.get_rect().width + stoppingGap
            x[direction][lane] += temp
        elif(direction=='down'):
            temp = self.image.get_rect().height + stoppingGap
            y[direction][lane] -= temp
        elif(direction=='up'):
            temp = self.image.get_rect().height + stoppingGap
            y[direction][lane] += temp
        simulation.add(self)

    def render(self, screen):
        screen.blit(self.image, (self.x, self.y))

    def move(self):
        if(self.direction=='right'):
            if(self.crossed==0 and self.x+self.image.get_rect().width>stopLines[self.direction]):   # if the image has crosse
                self.crossed = 1
            if((self.x+self.image.get_rect().width<=self.stop or self.crossed == 1 or (currentGreen==0 and currentYellow==0))
            # (if the image has not reached its stop coordinate or has crossed stop line or has green signal) and (it is eith
                self.x += self.speed  # move the vehicle
            else:
                self.idle_time += 1  # Increment idle time when the vehicle is not moving

        elif(self.direction=='down'):
            if(self.crossed==0 and self.y+self.image.get_rect().height>stopLines[self.direction]):
                self.crossed = 1
            if((self.y+self.image.get_rect().height<=self.stop or self.crossed == 1 or (currentGreen==1 and currentYellow==0)
                self.y += self.speed
            else:
                self.idle_time += 1  # Increment idle time when the vehicle is not moving

        elif(self.direction=='left'):
            if(self.crossed==0 and self.x<stopLines[self.direction]):
                self.crossed = 1
            if((self.x>=self.stop or self.crossed == 1 or (currentGreen==2 and currentYellow==0)) and (self.index==0 or self.
                self.x -= self.speed
            else:
                self.idle_time += 1  # Increment idle time when the vehicle is not moving
```

jupyter Amber_Green Model_01_11_2023-Copy1 Last Checkpoint: Yesterday at 5:34 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```python
        if((self.y+self.image.get_rect().height<=self.stop or self.crossed == 1 or (currentGreen==1 and currentYellow==0)
            self.y += self.speed
        else:
            self.idle_time += 1  # Increment idle time when the vehicle is not moving

    elif(self.direction=='left'):
        if(self.crossed==0 and self.x<stopLines[self.direction]):
            self.crossed = 1
        if((self.x>=self.stop or self.crossed == 1 or (currentGreen==2 and currentYellow==0)) and (self.index==0 or self.
            self.x -= self.speed
        else:
            self.idle_time += 1  # Increment idle time when the vehicle is not moving

    elif(self.direction=='up'):
        if(self.crossed==0 and self.y<stopLines[self.direction]):
            self.crossed = 1
        if((self.y>=self.stop or self.crossed == 1 or (currentGreen==3 and currentYellow==0)) and (self.index==0 or self.
            self.y -= self.speed
        else:
            self.idle_time += 1  # Increment idle time when the vehicle is not moving



    if self.crossed == 1:
        return  # Skip counting if the vehicle has already crossed

    if self.prt_countdown > 0:
        self.prt_countdown -= 1  # Countdown PRT timer
        return  # Vehicle does not move until PRT timer reaches zero

    if (
        (self.direction == 'right' and self.x + self.image.get_rect().width > stopLines[self.direction]) or
        (self.direction == 'down' and self.y + self.image.get_rect().height > stopLines[self.direction]) or
        (self.direction == 'left' and self.x < stopLines[self.direction]) or
        (self.direction == 'up' and self.y < stopLines[self.direction])
    ):
        self.crossed = 1
        lane_counters[self.direction] += 1
        if self.start_up_time == 0:
            self.start_up_time = timeElapsed

        if self.start_up_time == 0 and ((self.direction == 'right' and self.x > stopLines[self.direction]) or
                                        (self.direction == 'down' and self.y > stopLines[self.direction]) or
                                        (self.direction == 'left' and self.x < stopLines[self.direction]) or
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
# Initialization of signals with default values
def initialize():
    ts1 = TrafficSignal(0, defaultYellow, defaultGreen[0])
    signals.append(ts1)
    ts2 = TrafficSignal(ts1.red+ts1.yellow+ts1.green, defaultYellow, defaultGreen[1])
    signals.append(ts2)
    ts3 = TrafficSignal(defaultRed, defaultYellow, defaultGreen[2])
    signals.append(ts3)
    ts4 = TrafficSignal(defaultRed, defaultYellow, defaultGreen[3])
    signals.append(ts4)
    repeat()

def printStatus():
    for i in range(0, 4):
        if(signals[i] != None):
            if(i==currentGreen):
                if(currentYellow==0):
                    print(" GREEN TS",i+1,"-> r:",signals[i].red," y:",signals[i].yellow," g:",signals[i].green)
                else:
                    print("YELLOW TS",i+1,"-> r:",signals[i].red," y:",signals[i].yellow," g:",signals[i].green)
            else:
                print("   RED TS",i+1,"-> r:",signals[i].red," y:",signals[i].yellow," g:",signals[i].green)
    print()

def repeat():
    global currentGreen, currentYellow, nextGreen
    while signals[currentGreen].red > 0:  # while the red timer of current green signal is not zero
        printStatus()
        updateValues()
        time.sleep(1)

    currentYellow = 1  # set yellow signal on

    while signals[currentGreen].yellow > 0:  # while the timer of current yellow signal is not zero
        printStatus()
        updateValues()
        time.sleep(1)

    currentYellow = 0  # set yellow signal off

    while signals[currentGreen].green > 0:  # while the timer of current green signal is not zero
        printStatus()
        updateValues()
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
        time.sleep(1)

    # Reset all signal times of current signal to default times
    signals[currentGreen].green = defaultGreen[currentGreen]
    signals[currentGreen].yellow = defaultYellow
    signals[currentGreen].red = defaultRed

    currentGreen = nextGreen  # set next signal as green signal
    nextGreen = (currentGreen + 1) % noOfSignals  # set next green signal
    signals[nextGreen].red = signals[currentGreen].yellow + signals[currentGreen].green  # set the red time of next to next s

    repeat()




# Update values of the signal timers after every second
def updateValues():
    for i in range(0, noOfSignals):
        if(i==currentGreen):
            if(currentYellow==0):
                signals[i].green-=1
            else:
                signals[i].yellow-=1
        else:
            signals[i].red-=1

# Generating vehicles in the simulation
def generateVehicles():
    while(True):
        vehicle_type = random.randint(0,3)
        lane_number = random.randint(1,2)
        temp = random.randint(0,99)
        direction_number = 0
        dist = [25,50,75,100]
        if(temp<dist[0]):
            direction_number = 0
        elif(temp<dist[1]):
            direction_number = 1
        elif(temp<dist[2]):
            direction_number = 2
        elif(temp<dist[3]):
            direction_number = 3
```

```python
            direction_number = 3
        Vehicle(lane_number, vehicleTypes[vehicle_type], direction_number, directionNumbers[direction_number])
        time.sleep(1)

def simTime():
    global timeElapsed, simulationTime
    while(True):
        timeElapsed += 1
        time.sleep(1)
        if(timeElapsed==simulationTime):
            showStats()
            os._exit(1)

class Main:
    thread1 = threading.Thread(name="initialization",target=initialize, args=())    # initialization
    thread1.daemon = True
    thread1.start()

    # Colours
    black = (0, 0, 0)
    white = (255, 255, 255)

    # Screensize
    screenWidth = 1400
    screenHeight = 800
    screenSize = (screenWidth, screenHeight)

    # Setting background image i.e. image of intersection
    background = pygame.image.load('intersection.png')

    screen = pygame.display.set_mode(screenSize)
    pygame.display.set_caption("SIMULATION")

    # Loading signal images and font
    redSignal = pygame.image.load('red.png')
    yellowSignal = pygame.image.load('yellow.png')
    greenSignal = pygame.image.load('green.png')
    font = pygame.font.Font(None, 30)

    thread2 = threading.Thread(name="generateVehicles",target=generateVehicles, args=())    # Generating vehicles
    thread2.daemon = True
    thread2.start()
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
    thread3.start()

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()

        screen.blit(background,(0,0))   # display background in simulation
        for i in range(0,noOfSignals):  # display signal and set timer according to current status: green, yello, or red
            if(i==currentGreen):
                if(currentYellow==1):
                    signals[i].signalText = signals[i].yellow
                    screen.blit(yellowSignal, signalCoods[i])
                else:
                    signals[i].signalText = signals[i].green
                    screen.blit(greenSignal, signalCoods[i])
            else:
                if(signals[i].red<=10):
                    signals[i].signalText = signals[i].red
                else:
                    signals[i].signalText = "---"
                screen.blit(redSignal, signalCoods[i])
        signalTexts = ["","","",""]

        # display signal timer
        for i in range(0,noOfSignals):
            signalTexts[i] = font.render(str(signals[i].signalText), True, white, black)
            screen.blit(signalTexts[i],signalTimerCoods[i])


        # display the vehicles
        for vehicle in simulation:
            screen.blit(vehicle.image, [vehicle.x, vehicle.y])
            font_idle_time = pygame.font.Font(None, 20)
            idle_time_text = font_idle_time.render("Idle Time: " + str(vehicle.idle_time), True, (255, 255, 255))
            screen.blit(idle_time_text, (vehicle.x, vehicle.y - 20))
            #screen.blit(vehicle.image, [vehicle.x, vehicle.y])
            #font_control_delay = pygame.font.Font(None, 10)
            #control_delay_text = font_control_delay.render("Control Delay: " + str(vehicle.control_delay), True, (255, 255,
            #screen.blit(control_delay_text, (vehicle.x, vehicle.y - 40))  # Adjust the position as needed

            vehicle.move()
        # Display lane counters
```