Supreme Optimizer

Griffin Donnelly

Donovan Edwards

Michael Yue

SANTA CLARA UNIVERSITY

Department of Civil, Environmental, and Sustainable Engineering
& Department of Electrical and Computer Engineering
& Department of Computer Science and Engineering

I hereby recommend that the
SENIOR DESIGN PROJECT REPORT
prepared under my supervision by

GRIFFIN DONNELLY, DONOVAN EDWARDS, and MICHAEL YUE
entitled
SUPREME OPTIMIZER

be accepted in partial fulfillment of the requirements for the degree of
BACHELOR OF SCIENCE IN
CIVIL, ENVIRONMENTAL, AND SUSTAINABLE ENGINEERING
& BACHELOR OF SCIENCE IN
ELECTRICAL AND COMPUTER ENGINEERING
& BACHELOR OF SCIENCE IN
COMPUTER SCIENCE AND ENGINEERING

Advisors: [illegible] Date June 5, 2020

yi fang 06/09/2020

Rong He 06/09/2020

Department Chairs: Date

[illegible] Jun 10, 2020

[illegible]
Shoba Krishnan (Jun 10, 2020 13:38 PDT) Jun 10, 2020

Nam Ling
Nam Ling (Jun 10, 2020 23:19 PDT) Jun 10, 2020

SUPREME OPTIMIZER

by

GRIFFIN DONNELLY,

DONOVAN EDWARDS,

&

MICHAEL YUE

SENIOR DESIGN PROJECT REPORT

submitted to

the Department of Civil, Environmental, and Sustainable Engineering

& the Department of Electrical and Computer Engineering

& the Department of Computer Science and Engineering

of

SANTA CLARA UNIVERSITY

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Civil, Environmental, and Sustainable Engineering

& Bachelor of Science in Electrical and Computer Engineering

& Bachelor of Science in Computer Science and Engineering

Santa Clara, California

Spring 2020

ACKNOWLEDGEMENTS

Melissa Chee

Transportation Electrical Engineer, CalTrans

Jonathan Chimento, Class of 2013

Traffic Engineer, San Francisco Department of Transportation

Dr. Yi Fang

Professor of Computer Science and Engineering, Santa Clara University

Dr. Rachel He

Professor of Civil Engineering, Santa Clara University

Dr. Sarah Wilson

Professor of Electrical and Computer Engineering, Santa Clara University

Abstract

A Python-based machine learning algorithm was designed for the function of the traffic signal controller. Traffic signal controllers are the decision-making component within a traffic control box. Signal controllers determine when and which traffic lights transition from one phase to the next. The deep Q-learning algorithm designed in this project looked to decrease average vehicle delay, the expected amount of stoppage the average vehicle should expect, by a minimum of 10%. This reduction in vehicle wait times will have a noticeable impact on vehicle emissions created by interrupted vehicle flow, making the signalized intersection system more environmentally friendly. On top of these performance-based metrics, the design provided by this project can be implemented theoretically in easily available and cost-reasonable hardware, allowing transportation authorities to save considerable amounts of taxpayer money on the designed product.

Table of Contents

Report Section	Page
Certificate of Approval…………………………………………………………………..	i
Title Page……………………………………………………………………………...…	ii
Acknowledgements…………………………………………………………………....	iii
Abstract……………………………………………………………………………….	iv
Table of Contents……………………………………………………………………..	v
Introduction……………………………………………………………………….....	1
Alternative Analysis…………………………………………………………………..	5
Design Criteria and Performance……………………………………………………..	11
Software Design……………………………………………………………………...	12
Summary Description of Designed Facilities………………………………………....	20
Cost Estimate…………………………………………………………………………	23
Conclusion……………………………………………………………………………	24
References…………………………………………………………………………....	26

Appendixes

A. Engineering Design Documents For Campbell and El Camino Real Intersection
B. HCM and Webster's Model Specification
C. Code

1. Introduction

1.1 Project History:

The initial concept of the Supreme Optimizer project began with a small research project between Professor Rachel He and Griffin Donnelly. The idea of optimizing signal phase lengths based on evolving traffic flows was put forward, and the project was addressed as interdisciplinary beginning in the Spring quarter of 2019. Michael Yue, representing the Computer and Electrical Engineering Departments, was invited to work on the project by Griffin. A third team member, Donovan Edwards, was brought onto the project from the Electrical Engineering Department. During the summer, design brainstorming simplified the project deliverable to the signal controller, the device in traffic control boxes that determines when to change the traffic lights from green, yellow, or red. These components of traffic engineering will be discussed in further detail in the design criteria section of the document.

1.2 Project Objective:

The purpose of this Senior Design Project was to redesign the traffic signal controller to reduce average vehicle delay from current conditions benchmark and decrease the physical size of the controller device itself. The device was expected to decrease average vehicle delays at all intersections in the study by a minimum of 10%. The final design deliverable was also to bring forward a level of adaptability and predictability using collected statistical data on vehicle volumes, to adjust and determine the phasing for green, yellow, and red signals in each direction.

The original project objective was to implement the controller in both hardware and software. As the project evolved the team focused on the modeling, system design and software component of the project. The project software and simulation is portable across multiple intersections and various traffic conditions. Future work for this project could include applying the software to an embedded traffic system. This would likely include writing firmware and developing further hardware to sense cars and the traffic environment as a whole. An optimized version of the software algorithm can also be explored for future works. The software and simulation based objectives were achieved by reviewing past traffic theory, applying modern computational advancements to their purpose, and using these advancements to reduce key traffic metrics such as mean vehicle wait time.

1.3 General Site Description & Scope:

The project was focused near Santa Clara University, but the final deliverable design is scalable and expected to adapt to all standard street intersections. With that said, initial data collection and field observations were done on three local intersections. These intersections, as shown in the Figure below, are from bottom to top: Alameda & Newhall, Alameda & El Camino Real, and El Camino Real & Campbell **[Figure 1]**.

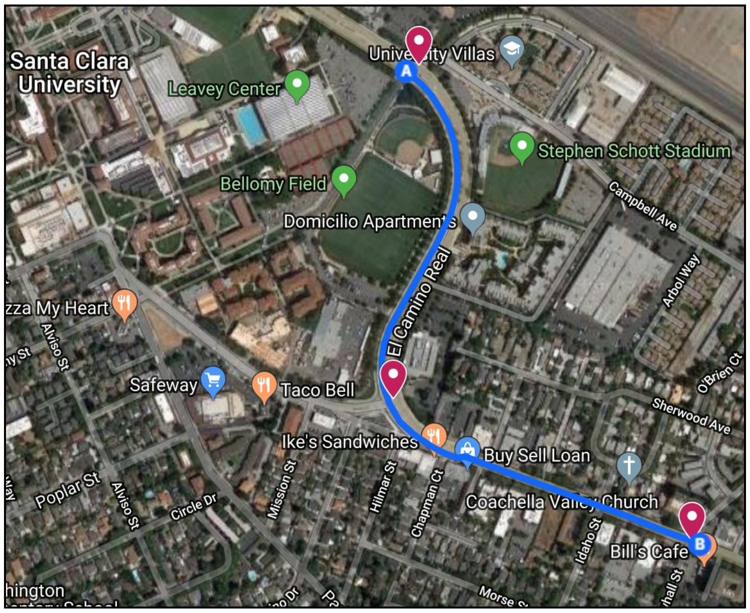


Figure 1: Map displaying local study intersections with crimson markers.

The current traffic control systems near Santa Clara University use actuated signal controls, and in some cases, include coordination as well. Actuated-coordinated plans, those used on Alameda & El Camino Real and El Camino Real & Campbell, determine green light times based on the arrival of cars on the detector system. Coordinated signal controllers also align their green phases along the major arterial, in this case El Camino Real, in order to minimize stoppage for large traffic platoons. Uncoordinated signal systems have the same actuation features as coordinated signals, but do not align their traffic platoons with other intersections in the corridor.

Existing traffic operations for signal controllers are outlined in traffic signal control plans. These plans, seen in [**Figure 2]**, state the lengths of time allocated for the different phases of movement at the intersection. These phases are directly connected to the available turning movements at an intersection. Similar to a Mealy finite state machine, the phases can be represented as states and the inductor loops can be represented as the inputs to the state machine. The state machine will only change states if certain predetermined conditions are met in the time domain and the traffic is in the necessary state.

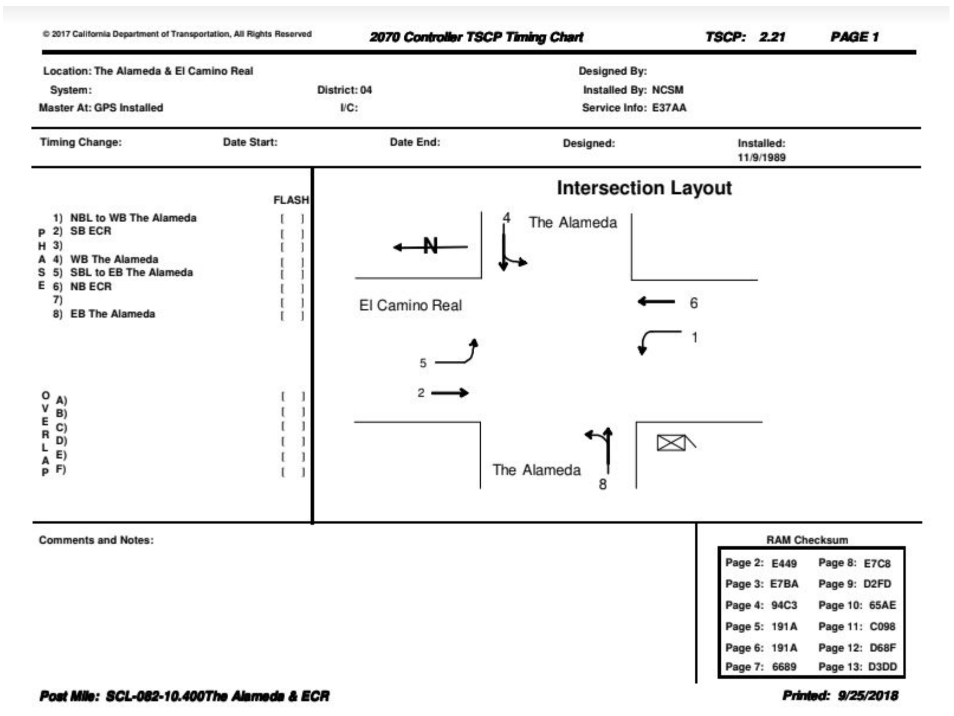
In the documentation in **[Figure 2]**, each turning movement is given a number representing a phase. The documentation in **[Figure 3]** contains information regarding the lengths and different phase conditions. The phase timing sheet displays several time variables, all of which are in the time units of seconds. The following table displays the variables and their corresponding definitions:

Table 1: Phase timing variables and definitions

Extension	Vehicle passes over a detector, time is added if it surpasses minimum green
Maximum Gap	Largest value of green time extension possible, due to several vehicle detections in rapid sequence
Minimum Gap	Lowest possible time extension allowable, maximum gap gradually decreases to this through “Reduce Gap By”
Reduce Gap By	Amount of time reduced from “Maximum Gap” for allowable extension

Reduce Every	Amount of time that must pass before "Reduce Gap By" increments.

The sum of all phase lengths is called a cycle, a value determined by the HCM and Webster's equations in **Appendix B**. Phase signal documentation for El Camino Real and Campbell is located in **Appendix A**.



© 2017 California Department of Transportation, All Rights Reserved

2070 Controller TSCP Timing Chart ***TSCP: 2.21*** ***PAGE 1***

Location: The Alameda & El Camino Real		Designed By:
System:	District: 04	Installed By: NCSM
Master At: GPS Installed	I/C:	Service Info: E37AA

Timing Change:	Date Start:	Date End:	Designed:	Installed: 11/9/1989

PHASE		FLASH
1)	NBL to WB The Alameda	[ ]
2)	SB ECR	[ ]
3)		[ ]
4)	WB The Alameda	[ ]
5)	SBL to EB The Alameda	[ ]
6)	NB ECR	[ ]
7)		[ ]
8)	EB The Alameda	[ ]

OVERLAP	
A)	[ ]
B)	[ ]
C)	[ ]
D)	[ ]
E)	[ ]
F)	[ ]

Intersection Layout

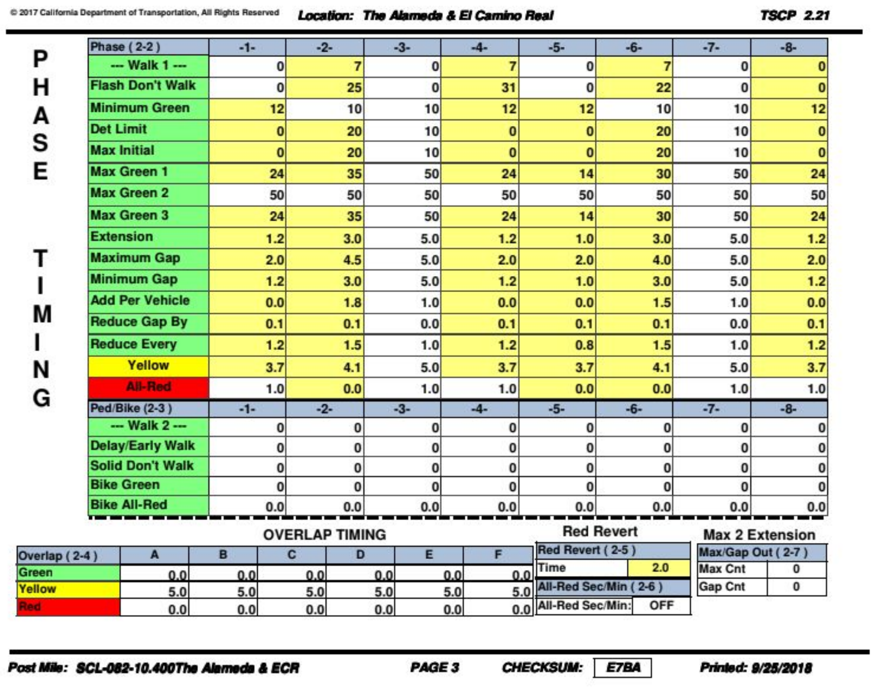
Comments and Notes:

RAM Checksum

Page 2:	E449	Page 8:	E7C8
Page 3:	E7BA	Page 9:	D2FD
Page 4:	94C3	Page 10:	65AE
Page 5:	191A	Page 11:	C098
Page 6:	191A	Page 12:	D68F
Page 7:	6689	Page 13:	D3DD

Post Mile: SCL-082-10.400The Alameda & ECR ***Printed: 9/25/2018***

Figure 2: Map displaying local study intersections with crimson markers.



© 2017 California Department of Transportation, All Rights Reserved **Location: The Alameda & El Camino Real** **TSCP 2.21**

PHASE TIMING

Phase (2-2)	-1-	-2-	-3-	-4-	-5-	-6-	-7-	-8-
--- Walk 1 ---	0	7	0	7	0	7	0	0
Flash Don't Walk	0	25	0	31	0	22	0	0
Minimum Green	12	10	10	12	12	10	10	12
Det Limit	0	20	10	0	0	20	10	0
Max Initial	0	20	10	0	0	20	10	0
Max Green 1	24	35	50	24	14	30	50	24
Max Green 2	50	50	50	50	50	50	50	50
Max Green 3	24	35	50	24	14	30	50	24
Extension	1.2	3.0	5.0	1.2	1.0	3.0	5.0	1.2
Maximum Gap	2.0	4.5	5.0	2.0	2.0	4.0	5.0	2.0
Minimum Gap	1.2	3.0	5.0	1.2	1.0	3.0	5.0	1.2
Add Per Vehicle	0.0	1.8	1.0	0.0	0.0	1.5	1.0	0.0
Reduce Gap By	0.1	0.1	0.0	0.1	0.1	0.1	0.0	0.1
Reduce Every	1.2	1.5	1.0	1.2	0.8	1.5	1.0	1.2
Yellow	3.7	4.1	5.0	3.7	3.7	4.1	5.0	3.7
All-Red	1.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0

Ped/Bike (2-3)	-1-	-2-	-3-	-4-	-5-	-6-	-7-	-8-
--- Walk 2 ---	0	0	0	0	0	0	0	0
Delay/Early Walk	0	0	0	0	0	0	0	0
Solid Don't Walk	0	0	0	0	0	0	0	0
Bike Green	0	0	0	0	0	0	0	0
Bike All-Red	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

OVERLAP TIMING

Overlap (2-4)	A	B	C	D	E	F
Green	0.0	0.0	0.0	0.0	0.0	0.0
Yellow	5.0	5.0	5.0	5.0	5.0	5.0
Red	0.0	0.0	0.0	0.0	0.0	0.0

Red Revert

Red Revert (2-5)	
Time	2.0
All-Red Sec/Min (2-6)	
All-Red Sec/Min:	OFF

Max 2 Extension

Max/Gap Out (2-7)	
Max Cnt	0
Gap Cnt	0

Post Mile: SCL-082-10.400The Alameda & ECR **PAGE 3** **CHECKSUM:** E7BA **Printed: 9/25/2018**

Figure 3: Phase timing sheet for the Alameda and El Camino Real intersection

2. Alternative Analysis

2.1 Simulation Program:

There are several computer programs that can simulate traffic networks, specifically those with signalized intersections. The two programs that will be compared in this document are Synchro's traffic software and Simulation of Urban Mobility (SUMO) software packages. Synchro is an industry standard in determining the serviceability of traffic infrastructure given a known volume of vehicles. Professionally, traffic engineers use Synchro and its reports in Level of Service (LOS) and average vehicle delay as submittals in a variety of traffic engineering projects. Synchro can simulate pre-timed, actuated, and actuated-coordinated signal timing plans for signalized intersections. Synchro develops a significant amount of the simulation internally, where SUMO requires detailed input and programming from the user's end. This means Synchro is incredibly simple compared to SUMO, whereas SUMO allows for a higher level of control over the simulated environment.

One of the biggest disadvantages of the Synchro traffic software is the inability to customize traffic flows. Synchro takes a large scale view, looking at average traffic flows on a per hour basis. The team was not able to figure out how traffic is simulated within this hour, however, the most reliable information the team could find showed a uniform distribution. This means Synchro uses the vehicles per hour number equally over a 60 minute period. This was not reflective of the traffic conditions, and for the purposes of this project was not sufficient. SUMO allows for the simulation of traffic on a minute by minute basis.

Using SUMO to simulate traffic was done by taking real traffic data. Accurate traffic data is necessary for proper traffic analysis in the traffic engineering field and this project. However, the group used the traffic data to create probability distribution functions which can create many realistic traffic scenarios. Optimizing a traffic controller for a single day is not useful, since each day has different traffic conditions. However, the different conditions still follow the same probability distributions.

Another key aspect to representing an intersection accurately was the incorporation of platoon data. Platoon data is a specific form of traffic observation that has the data collector group their vehicle counts in platoons, or clumps, of passing automobiles. This is done to increase the realism of the stochastic simulation, as traffic does not arrive singularly on major arterial roads. Each vehicle can be thought of having an inherent ‘stickiness’. This means that the less vehicles on the road, the more spread out they are. The more vehicles on the road, the more clumped up near each other they are. Platoons and how they are used in the simulation is further described in section 4.4.

In order to properly simulate the team's machine learning algorithm in virtual space, SUMO was the chosen program. SUMO also allowed for more detailed control of vehicle flow simulation, allowing platoon data to be properly utilized. Due to previous signalized intersections along the corridor, traffic arrives in these waves, a defining factor for coordinating intersections. By simulating traffic platoons, the Q-Learning algorithm will be able to handle large traffic platoon situations similar to how a coordinated signal plan would operate.

2.2 Evaluation Criteria:

There are several considerations that had to be made for the traffic signal controller component of the traffic control box. *Public safety* is the first priority. Safety can be measured by a variety of factors, including but not limited to hardware reliability, durability, and consistency. Controllers with consistent quality in electrical signal output are preferable and easier to work with than more advanced and complex controller systems with a higher likelihood of electrical component error or total failure.

From a public safety perspective, there are also considerations in selecting the controller model. The Highway Capacity Manual (HCM) model and Webster's equation can be compared on a public safety as well. The HCM model includes consideration for pedestrians, inherently making the model safer than Webster's equation. Webster's equation does not consider pedestrians in the phase length calculations. Either equation can be used for determining cycle length for each intersection, however both have fixed timing values in the controller. This includes fixed values for green minimum and green maximum times. The software of the project improved upon both of these previously mentioned models due to the variable in green maximum times. The software of this project still maintained safety by using the same green minimum times in order to protect pedestrians. However, pedestrians were not used in the simulation, in order to simplify the process and comparisons.

Responsiveness and processing speed are the second most important aspects of the component design, with relevance in both the hardware and software aspect of this project. In terms of hardware, responsiveness is relatively straightforward, as there are clear metrics in processing speed that are provided by the supplier. The software end becomes more tricky, as code, once complete, will have to be tested for optimal efficiency. The software side could not be tested for this project and was out of project scope, so this analysis only looked at the given hardware speeds.

Memory is important for storing and manipulating the data collected from the detector components connected to the controller. Capacity and durability play a factor in the hardware side of the component, ensuring enough data can be stored for weeks and possibly even months at a time in unusual conditions. Ease of access was also considered in this criteria, and if any hardware component was known to be difficult to integrate with Python, it scored lower in this category.

Cost was an obvious and important part of choosing the hardware. Although this is placed below most performance metrics, it should be noted that cost does put a ceiling on many of the options that were able to be pursued in this project. Given that the senior design project needed implementation at every traffic control box, its cost had to be less than that of current devices given the objective of improvement. Power usage can also be included in this aspect, as the most important effect of power usage will lie on the overhead cost that comes with supplying electricity to the system.

Numerically, the alternative solutions were ranked relative to each other. The criteria were weighed from zero to three with three being the highest level of importance and zero representing a category that did not apply to the given alternatives. The alternatives were evaluated based on the criteria from zero to five where five was the best demonstration of criteria, one was the minimal allowable for the criteria, and zero expressed that the alternative could not be judged by the criteria. Based on the total score calculated from these criteria for the alternatives, an alternative solution was chosen for this project.

2.2.1 Software Architecture:

There were several software implementation options for this optimization algorithm. Tensorflow is a machine learning Python library that can be used to predict traffic patterns and conditions on local roads. Tensorflow is a tool to predict the traffic pattern rather than optimize the pattern. Prediction is inherently riskier than optimization, but makes the program more responsive. Reinforcement learning is an optimization algorithm that sets rewards and punishments for certain situations. Reinforcement learning focuses on minimizing wait time of cars at an intersection. Based on formulas, delay time is calculated for each situation, and a rewarded situation will occur more often than a punished situation. Gradient descent is an optimization technique to find a local minimum using approximation integrals. Gradient descent will generate a signal plan and the live traffic data will make the system dynamic. Gradient descent can also optimize for multiple traffic variables at an intersection. Dynamic programming uses a state machine and decision tree to test the best results from a list and optimize the best stages for a signal plan. Q-learning is a heuristic approach that follows a mathematical formula to find the best timings based on risks, rewards, and consequences. From the analysis, the team implemented Q-learning into the software.

The project team wanted to ensure that their traffic-light algorithm and its software implementation would not cause a traffic accident. This is important both conceptually and in testing because bad performance affects the metrics of evaluation for the team's tests. Responsiveness is also important in software because the optimization algorithm needed to be efficient and effective. Memory is not as important because the methods to handle memory were similar across all alternatives. Finally, cost is not important because the team used all free and open source programs to develop and test the system. All of the criteria is listed in the alternative analysis matrix below **[Table 2]**.

Table 2: Analysis Matrix for software alternatives.
Dynamic Programming and Q-learning chosen as most preferred alternatives.

	Tensorflow	Reinforcement Learning	Gradient Descent	Dynamic Programming	Q-learning
Safety (3)	2	2	3	4	3
Responsiveness (2)	3	3	1	3	3
Memory (1)	3	3	3	2	4

Cost (0)	0	0	0	0	0
Total	15	15	14	20	19

2.2.2 Cycle Length Modeling:

The two equations, Webster's and HCM cycle equations, are fundamental for traffic signalization and are tasked with achieving the same result, setting green time for a standard signal cycle length. See **Appendix B** for elaboration of Webster's and HCM cycle equations. The main consideration to have when comparing these two equations is their ability to be reliably implemented in a software based simulation. This means that a concise, fluid equation with fewer additional factors and variables makes simulation easier. In this scenario, Webster's equation was clearly simpler and preferable to the HCM model. Webster's equation does not require the conversion of traffic volumes to critical units. Critical units are a term used in the Highway Capacity Manual to describe a baseline "through" unit. Left-turn traffic and right-turn traffic need to be converted to through traffic in HCM, which is not the case in Webster's equation. Webster's equation considers the difference between left, through, and right when finding the ratio used to tabulate the actual vehicle volume over the saturation flow rate.

Cycle Length Modeling also has no cost comparison, as neither method is more expensive than the other. Accuracy and safety are the only measurable differences. Accuracy in this scenario also related to average vehicle delay and how well the model accommodates traffic flow patterns. Given that the HCM model is newer and a product of more in-depth data and empirical observation, it is generally more accurate than Webster's equation. For this reason, HCM was determined to be safer and the better choice for modeling the appropriate cycle length for signalized intersections, see **[Table 3]** for analysis matrix.

Table 3: Analysis Matrix for cycle length model alternatives. HCM Model chosen as most preferred alternative.

	Webster's Model	**HCM Model**
Safety (3)	4	5

Responsiveness (2)	3	3
Memory (0)	0	0
Cost (0)	0	0
Total	18	21

This project chose to focus on Q-learning for the software architecture component of the project. The main reason for this is because the Q-learning algorithm doesn't need to know the model of the environment. Q-learning also serves as a basis for deep Q-learning where the number of states action combinations can be approximated using a neural network. With dynamic programming, the number of states would be too large to process. The HCM Model was chosen as the method of establishing current conditions because this is the method currently used in the field. The project was provided with up-to-date signal timing plans, all of which are adjusted from original calculations using the HCM Model. The HCM Model also considers pedestrian traffic, which means it is inherently safer than Webster's equation.

3. Design Criteria and Performance Requirements

3.1 Level of Service:

The standard metric for determining the performance of a signalized intersection is level of service (LOS). This metric is an abbreviated version of average vehicle delay. Average vehicle delay is determined by summing the total delay experienced by all vehicles, in seconds, due to a traffic signal and dividing it by the total vehicle count that passes through the intersection.

$$D = \frac{C*[1-(g/C)]^2}{2*[1-(g/C)*X]}$$ **[Equation 1]**[1]

D = Delay (seconds/vehicle)

C = Cycle Length (seconds)

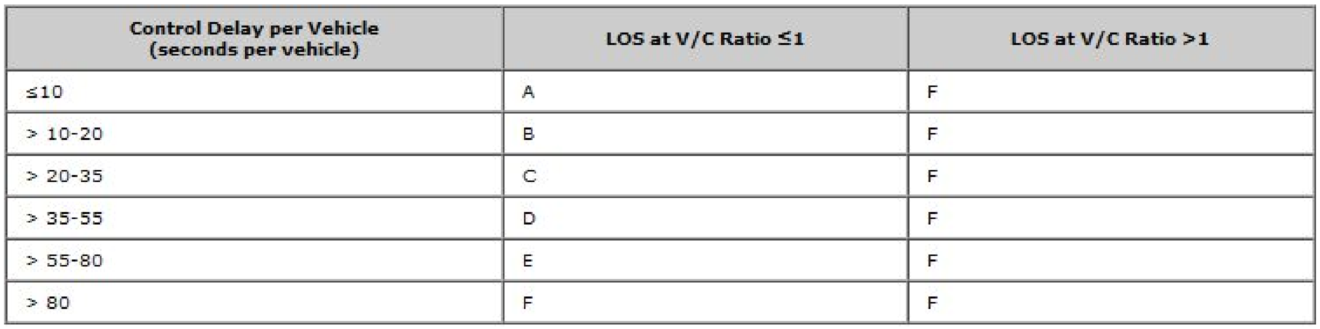
g = Effective green time (seconds)

X = volume/capacity ratio (same as “v/c” in HCM Model equation of **Appendix B**, usually 0.95)

The LOS metric simplifies this value into a letter grade form. Equivalent to the grading scale for academic work, LOS letter grades are done in descending order of serviceability from ‘A’ to ‘F’, skipping ‘E’. Table 4, below shows the average vehicle delay range for each letter.

Table of LOS Letter Grades and their corresponding Average Vehicle Delay time intervals [https://safety.fhwa.dot.gov/intersection/conventional/signalized/fhwasa13027/ch7.cfm]

Table 4: LOS table showing delay and letter grades



Control Delay per Vehicle (seconds per vehicle)	LOS at V/C Ratio ≤1	LOS at V/C Ratio >1
≤10	A	F
> 10-20	B	F
> 20-35	C	F
> 35-55	D	F
> 55-80	E	F
> 80	F	F

3.2 Average Vehicle Emissions:

Reducing vehicle emissions was a primary concern in regards to the sustainability aspect of the project. Although it has been shown that reduced vehicle delay decreases emissions, an important aspect of using the SUMO software program is the access to a variety of open-source tools that can measure different aspects of the simulation. The key metric of this project was reducing average vehicle wait time. SUMO has some vehicle emission measurement functions, but it was unclear how those functions calculated emissions. For this reason, the decision was made to focus on reducing vehicle wait times. The calculation for average wait time was reliable and apparent how the calculation was being accomplished. The results of the project assume that vehicle emissions will be reduced by lower wait times. It is currently unclear by how much a 10% reduction in vehicle wait times will realistically reduce emissions.

4. Software Design

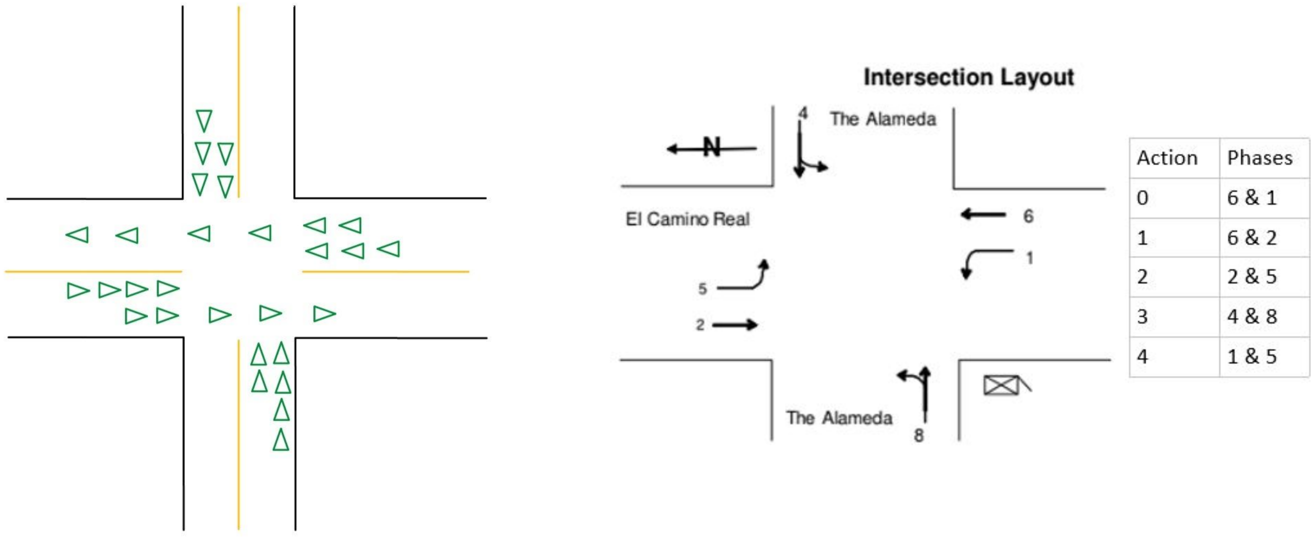
4.1 Overview:

The Supreme Optimizer project had three key components. The first component of the project was gathering traffic data from around the university. The second component of the project involved modeling the traffic appropriately to represent the intersections discussed accurately. The third component involved comparing a machine learning based traffic controller to the current method Caltrans uses to control intersections. This resulted in an estimated 10% increase in efficiency.

4.2 Q-Learning:

The team decided to implement a deep Q-learning algorithm in the traffic controller. Deep Q-learning is based on the traditional Q-learning algorithm with an additional deep neural network. The team decided to use deep Q-learning because of the effectiveness of the algorithm for a traffic environment. This method will give accurate and meaningful results while saving time on the simulation.

Q-learning is based on a reinforcement learning model to make the best decision based on a number of parameters.[2] Those parameters have to be defined appropriately for a traffic intersection. The team followed the equation in **[Figure 4]** for a traffic intersection. The Q-value is a function of state and action values that get stored and updated in a Q-table. Based on old Q-values, a new one is generated to represent the best action to take from a given state. This value determines which traffic light phase is the best to switch to for a given action. For a traffic intersection, the state refers to the number of cars at an intersection.[3] A snapshot of the intersection can be thought of as the state. Position, speed, velocity, and direction of each vehicle contributes to the state environment of the intersection. All of this information gets translated into a matrix value for the algorithm. In **[Figure 5]** an example of the state is shown to represent the intersection. The number of cars in each lane at an intersection can result in thousands of state combinations. From any state, one of five actions can be taken. The action refers to the traffic phase that is activated to turn green. Five unique actions are defined in **[Figure 5]**. Each action is a combination of traffic light phases activated for an intersection. The Q-table is a matrix that holds a specific value for any state action pair. An example of a basic Q-table setup is shown in **[Figure 6]**. When the Q-learning algorithm is executed, the Q-values in that Q-table get updated appropriately. Based on the learning process in the algorithm, a best action can be selected for a given state.



Action	Phases
0	6 & 1
1	6 & 2
2	2 & 5
3	4 & 8
4	1 & 5

Figure 4: State Representation(left), Action Definition(right)

The next parameter is reward. Reward is defined as the parameter that the system uses to judge consequences of actions taken. In the project, the reward is cumulative wait time of vehicles at an intersection. The wait time determines if an action was good or bad. Reward can also be thought of as an indicator to how well the algorithm performs. The discount factor is the parameter between 0 and 1 that determines the importance of future rewards compared to the immediate reward. The learning rate determines how much old information is replaced with the new information at every iteration. The learning rate is set to 0.002 in order to use most of the old information and thus learning slowly. All of these parameters are used in the Q-learning algorithm which is the basis of the project design.

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

Figure 5: Q-learning algorithm

4.3 Deep Q-learning:

In regular Q-learning, there needs to be a value for every state action pair. For a traffic intersection, this number is too large to manage in a controller. Instead, the Q-value is approximated from a given state using a deep neural network(DNN).[4] This method has the advantage of accurately predicting the Q-value from any state while decreasing the overhead needed with a Q-table. The team followed a framework to implement a basic neural network for deep Q-learning.[5] The states of the intersection are the input to the neural network. The DNN is based on the Q-learning equation and uses all the parameters to determine Q-values from every possible action. **[Figure 6]** shows the basic structure of the neural network and how the approximation of Q-values can be determined from a given state.[4] Based on the position, velocity, and speed of the vehicles at the intersection, the neural network can collapse this information into one Q-value. The difference between normal Q-learning and deep Q-learning is also shown in **[Figure 6].** In normal Q-learning, the Q-table determines the Q-value based on state and action pairs. In deep Q-learning, the neural network itself can determine the Q-value based on the state and approximate the Q-learning equation.

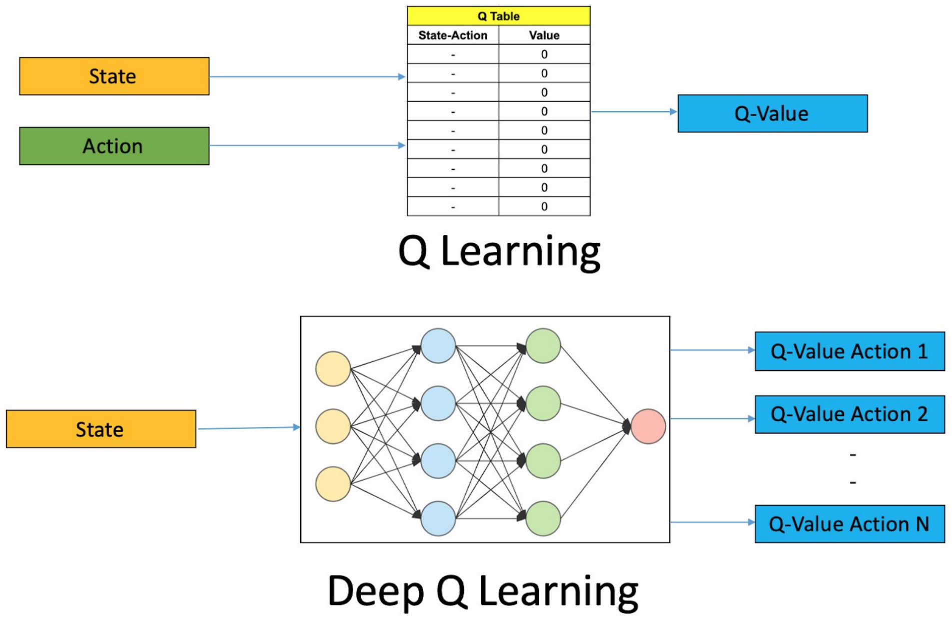


Figure 6: Q-learning vs Deep Q-learning

4.4 Implementing Traffic Data:

The team gathered traffic data at three intersections, El Camino Real (ECR) and Newhall, ECR and Alameda, and ECR and Campbell. This data was broken into tables as follows per each day of data collection. NT means north through, NL means north left, NR means north right, ST means south through and so on until ER for east right.

Table 5: Traffic volume table

Time	NT	NL	NR	ST	SL	SR	WT	WL	WR	ET	EL	ER
7:30 AM												
7:31 AM												
(7:32 ... … 8:58)												
8:59 AM												
9:00 AM												

The blank boxes represent the number of cars observed in each minute block of time. This number can vary significantly, and a link to the drive will be included in the appendix to the data for reference.

The traffic flow per minute data can be used to generate probabilities of how often that amount of cars will go through the intersection in a given minute. NT of all three intersections consistently had a vast majority of the traffic flow of each intersection (for reference this is coming out of the I-880 and going north on ECR towards 101-N). The percentage of traffic of NT compared against all traffic flow is not relevant, but it was just an apparent observation throughout the course of the project.

The lack of traffic flow in each of the other 11 directions could be modeled with a simple probability distribution (PDF). For the project's purpose, a simple PDF refers to the fact that the 11 minor distributions were normalized over the entire time period (7:30 AM to 9:00 AM). The last direction, NT, needed to be broken up into multiple different PDFs in order to accurately represent the intersections. The following is a graph of the Alameda and ECR intersection's NT traffic data on October 29, 2019.

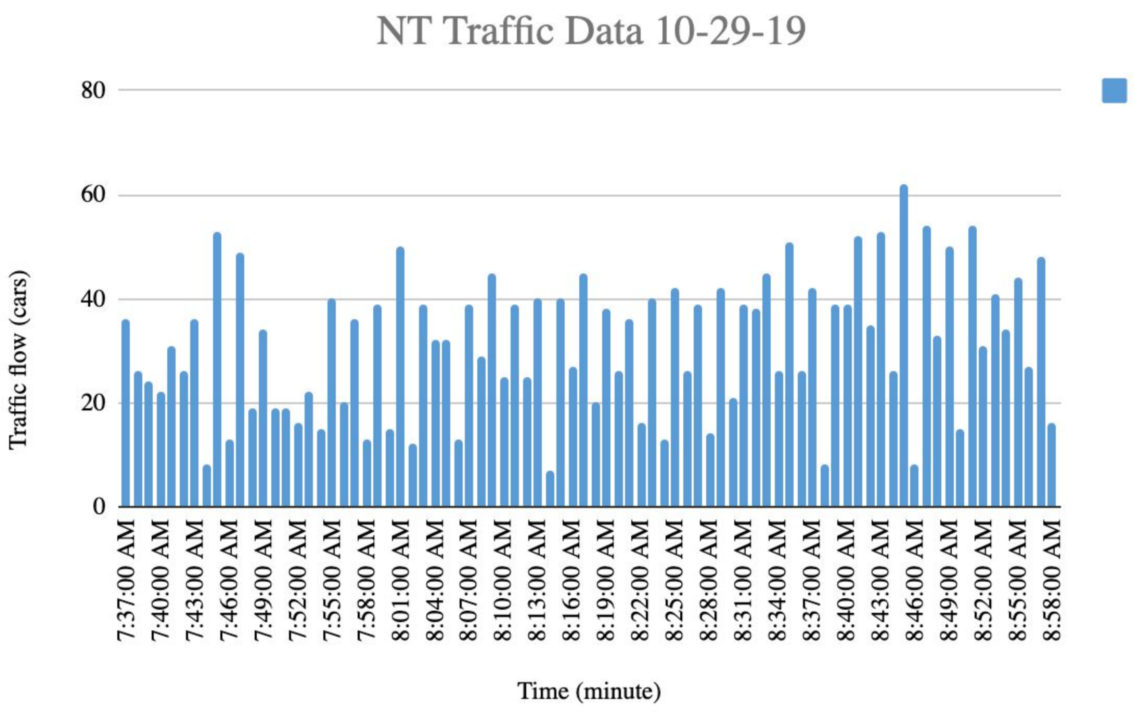


Figure 6: Traffic volume distribution

The NT traffic goes through either a middle process or a peak-valley process. A middle process refers to when the number of cars in a given minute is a middle amount (for this graph is approximately between 18-40) for two minutes back to back. Observations of this process occur around 7:40 AM and 8:04 AM. A peak-valley process refers to when the number of cars in two consecutive minutes has a spike (40< cars in a minute) followed by a decline (<18 cars in a minute). Peak-valley processes occur at 7:43 AM and 8:01 AM for reference.

Now that these processes have been defined, the probability of each occurring can be found. However, the traffic data needs to be broken up further. The probabilities of middle versus peak-valley changes significantly over the course of an hour and a half so multiple PDFs need to be created for different time periods. Note: each PDF was created using the best fit out of an exponential, gaussian, or poisson distributions.

For the project, traffic was divided into three separate PDFs in the following time periods: 7:30 AM -8:00 AM, 8:00AM-8:30AM, and 8:30AM-9:00AM. In each time period the following probabilities and PDFs were created:

P[peak-valley occurs] = 1 - P[middle occurs]; determines which PDF will be called
PDF of middle, PDF of valley, PDF of peak
Either middle will be called twice, or peak and then valley will be called. The reason for this order (peak then valley) is that over a long sequence it doesn't matter which is called first, as long as two peaks and two valleys are not next to each other. Picking an order up front ensures the two peaks or two valleys case cannot occur.

Now the problem arises of how to call those cars into existence within a given minute. This is where the team had to innovate a solution called platoon data. Platoon refers to the fact that as more cars show up together within a given minute the more likely the vehicles are to clump together. This means that another unique PDF needs to be created, however, this does not need to be broken into 3 time sections. Platoon distributions apply across time, since cars have an inherent "stickiness" quality, which means that cars tend to attract other cars at consistent rates.

Here is an example of a platoon PDF for ECR and Alameda.

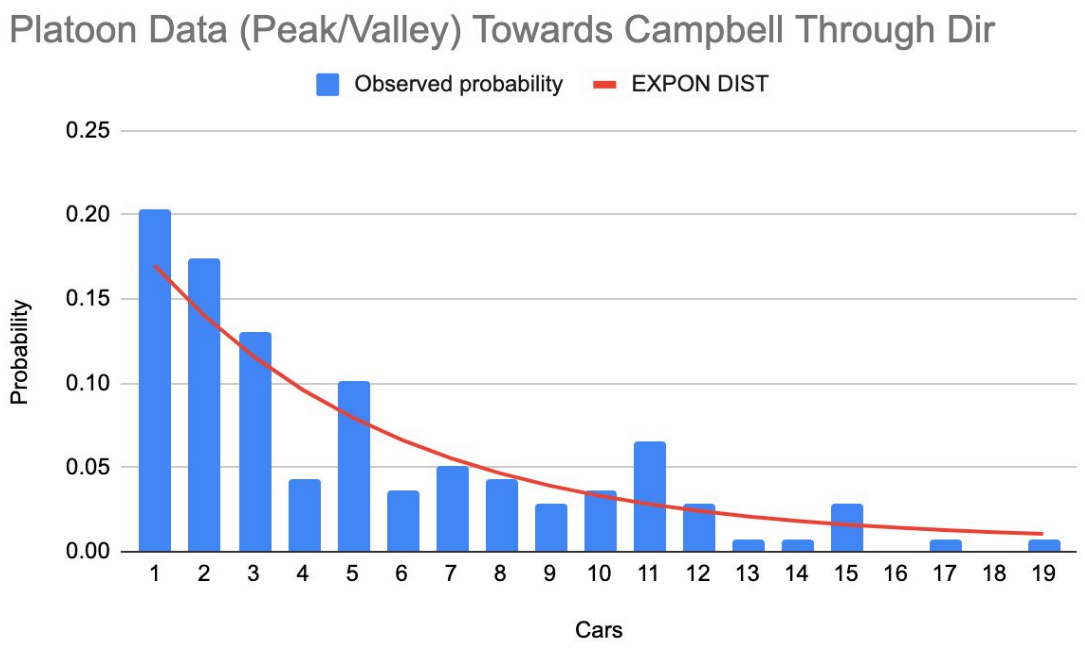


Figure 7: Platoon data

Platoons were measured by counting cars continuously flowing until a break was met. Then the platoon distribution is normalized in comparison to the total amount of cars flowing in a given minute, determined by the peak-valley and middle PDFs.

All of the PDFs were accessed by creating a random number, and assigning regions from 0-1 to determine vehicle calls and platoon size. All PDFs were stored in csv files, and all scripts were written in python. Vehicles are called into SUMO using XML files.

Here is a decision making process of the process of calling cars into a simulation in a descriptive method at a high level for clarity:

Traffic distribution creates a total number of cars in a 1 minute period.
The platoon distribution determines how large the clumps of cars are within the 1 minute period.
The platoons are then assigned times to appear at the intersection within the 1 minute period.
Repeat for the next minute long period.

5. Summary Description of Designed Facilities

5.1 Collecting Field Data:

The primary information the device will use for determining when to change signal phases was directional vehicle volumes. Directional vehicle volumes are traffic counts for each cardinal direction, broken down into their turning movement in either the left, through, or right turns. **[Figure 8]** displays the exact device used to collect traffic data in the field. Observationally, lanes are not distinguished outside of these turning movements. Data collected for a two-lane through volume is split equally, meaning that the volume for a direction is divided by the number of lanes supporting it. This is an assumption made for this project. Directional volume data acted as one of two major inputs for the signal phase equations, and behaved as the primary factor in ensuring the team's software was predictable, using known time data from previous but similar contexts to understand how long a green light should remain green. Additionally, field observations included vehicle platooning, where groups of cars were counted together as platoons in order to increase the accuracy of the simulated environment. By recording platoon counts, the team was able to generate vehicles in the SUMO environment that closely resemble actual traffic flows on the road.

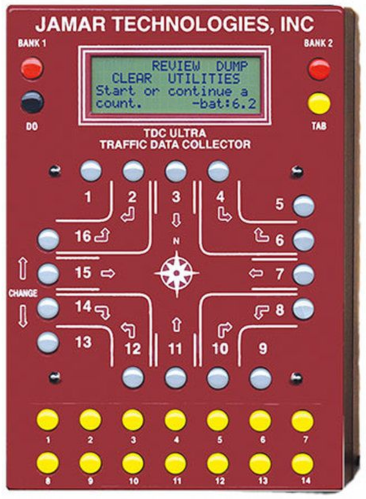


Figure 8: Jamar Technologies traffic counter.

5.2 Constructing the SUMO Simulation Environment:

To simulate the Q-learning and signal phase plans, the team used an open source software called Simulation of Urban Mobility (SUMO). SUMO can simulate any traffic environment through its programmable interface and native Python support. The Traci library in SUMO allows for unique traffic light situations such as a deep Q-learning model. The team used the various functions in Traci to determine the state, action, and reward values from the traffic environment. The deep Q-learning method written in Python was also integrated to interact with the SUMO simulation to generate results. The team also used SUMO to simulate the static phase plan. For this method, the team programmed the traffic lights to cycle through the signal plan to represent how the intersection currently operates.

Each intersection has a specific geometry that determines the signal phasing and directional movements of the incoming vehicles. These geometries include: lane count, left turn pockets, and lane directions. Once the geometry was established, signal phases had to be established for each group of non-conflicting vehicle movements. Directional movements like North-through and North-left can have coinciding green signals, as their vehicle patterns do not overlap. The same can be said for North-through and South-through, North-left and South-left, etc. These pairings can be seen in an example in the right image of **[Figure 4]**. These phasings were an important aspect of both the existing and project-designed signal control, as they represented the “choices” that the controllers will have to choose from in order to service the intersection. In the case of the existing signal controller, when using an actuated-coordinated phase plan, the phases must be serviced in a certain order. In the machine-learning based algorithm designed in this project, the phases were serviced in any order deemed necessary. It was expected that the same coordinating effect will take place between intersections because every controller down the road will eventually learn to recognize the ‘choices’ of the previous controller based on incoming traffic data.

5.3 Generating Traffic Flow in Simulated Environment:

After the geometry of each intersection was constructed virtually, vehicles had to be generated in real-time to test the signal controller's quality of service. As mentioned in Section 4.1, field data collected both vehicle per hour per lane counts, as well as probability distributions for platoon size based on observation. Using this information, vehicles can be generated on a per lane, per direction basis. The traffic flow model was stochastic, meaning that the vehicles enter the system in a seemingly random manner, but are constrained to the set parameter of platoon size and total vehicle flow for the peak hour. This can be seen in **Appendix C.**

4.4 Deliverables:

Once the simulated environment was complete, the neural network was trained on 2000 episodes. The training results can be seen in **[Figure 9].** Due to time, only the first 220 episodes are displayed in the results. The results show that initially, the data may suggest a high average wait time, but the average is expected to decrease as the episode lengths increase. On longer simulation episodes, a strong accuracy and better results can be expected to improve wait time. However, on the first 220 episodes, the average wait time is starting to decrease and is expected to stabilize under 20 seconds. After training the neural network, the system is tested by inputting any state to the system and generating a Q-value from the network. This is also done in simulation to control the lights at an intersection. To compare against the current static signal phase plan, a distribution of around 3000 cars was generated to run through the intersection. The team compared the wait time using the neural network and the static phase plan and calculated the average vehicle wait time. The team measured average vehicle wait time because it is the reward value for the Q-learning algorithm, and because it is the metric to directly determine level of service. From the deep Q-learning method, the level of service calculated was a B and the level of service using the signal phase plan was a C. Overall, from the initial results, this is about a 10% improvement in average vehicle wait time.

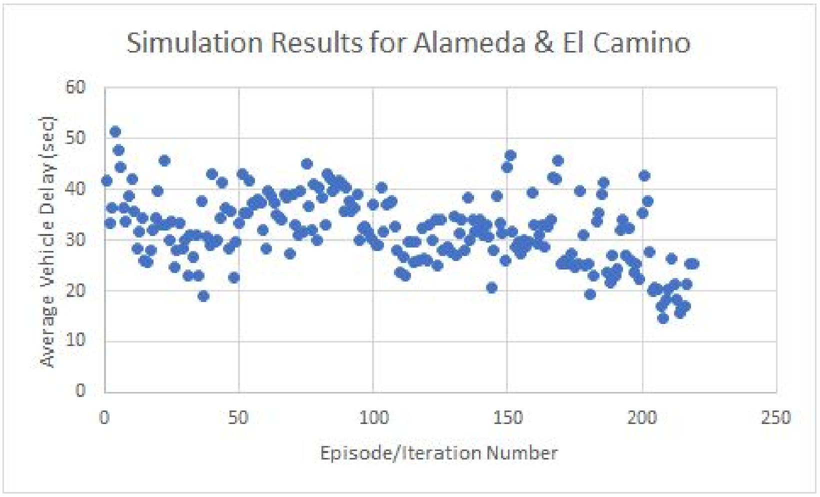
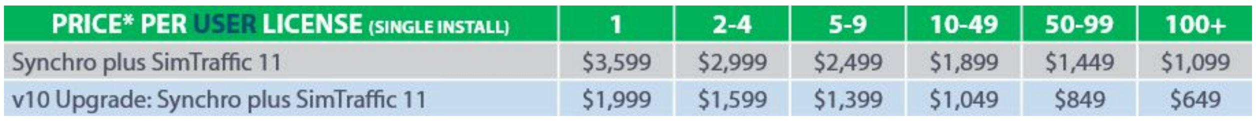


Figure 9: Deep Q-learning simulation results

6. <u>Cost Estimate</u>

The estimated cost of this project was difficult to set as confidently as most other Civil Engineering-related projects as this product was almost entirely within the bounds of open-source software. This project required no funds for the SUMO aspect of the project, but simulation was done for validity in Synchro and SimTraffic. The license for one copy of this program is $3,599, which scales down based on licenses purchased, see **[Table 6]**.

Table 6: Prices per license from https://www.trafficware.com/synchro-store.html



PRICE* PER USER LICENSE (SINGLE INSTALL)	1	2-4	5-9	10-49	50-99	100+
Synchro plus SimTraffic 11	$3,599	$2,999	$2,499	$1,899	$1,449	$1,099
v10 Upgrade: Synchro plus SimTraffic 11	$1,999	$1,599	$1,399	$1,049	$849	$649

Cost can also be estimated through the physical controller itself. Traffic signal controllers cost approximately $10,000 to upgrade to current standards, based on estimates from the U.S Department of Transportation.[6] A microcontroller that can profficiently run the project's machine learning algorithm for signal control would cost no more than $900, combined with $2,500 estimated for casing and wiring costs leaves the final price of a custom-built hardware system at roughly $3,400. This cost is significantly less than the current pricing for industry-standard signal controllers, which would save transportation departments thousands of dollars per replacement, not including the thousands of dollars saved in a self-correcting machine learning algorithm.

The cost of labor still had to be factored into account for the design, given that the high cost of signal controller systems is determined by the cost of research and development for the product's software and hardware setup. Working for an estimated 25 weeks, at an average of 15 hours per week per person, calculates to 1125 hours of work put towards this software design. Between the hourly wage of a traffic engineer and software engineer in the Bay Area, the hourly cost of work comes to approximately $45 dollars per hour, which means that the total cost of labor is $50,625. If the cost of the controller were equal to that of its industry-standard, only about eight would need to be sold to compensate for labor. This number would already put the project in the positive in terms of income, with the number of units being sold as incredibly small in comparison to the volume demand for traffic controller systems. For example, the City of San Jose alone has "over 900 signailzed intersections" in its jurisdiction.[7] Far more than eight of these units could be sold, further bringing down the cost to cover the initial investment.

Table 7: Summary of calculations

Avg hours per week	Avg weeks per year	Total hours per year	Hourly rate	Total project cost	Avg controller cost	Cost of Goods Sold	Profit per controller	8*Profit	Project income after 8 sales
15 hours	25 weeks	1125 hours	$45	$50,625	$10000	$3400	$6600	$52800	$52800- $50625 = $2175

7. Conclusion

Throughout this project, the team was able to develop a simulation model for deep Q-learning based implementation of a traffic light controller. The team used this implementation on a simulated environment representing various intersections around Santa Clara University. The team also determined the distribution of traffic at these specific intersections by conducting a live traffic study at these sites. After running those distributions through simulations, the deep Q- learning model gave us optimized average wait times of 14-16 seconds per vehicle. This was accomplished with a traffic light controller that adapted to the traffic environment and conditions. The team compared these results to the average wait times from a simulation that represented the current method of controlling traffic. These signal phase plans were run in simulation with the same traffic distributions at each intersection. The static simulation resulted in an average wait time of 20-25 seconds per vehicle. This results in about a 10% improvement in wait time by using the deep Q-learning model. Overall, the project group was able to develop a better solution to traffic light management at intersections around Santa Clara.

9. References

[1] “Traffic Signal Timing Manual,” *Traffic Signal Timing Manual: Chapter 4-6 - Office of Operations*.

[2] “Q-learning - Wikipedia.”

[3] X. Liang, X. Du, G. Wang, and Z. Han, *Deep reinforcement learning for traffic light control in vehicular networks. arXiv:1803.11115* .2018.

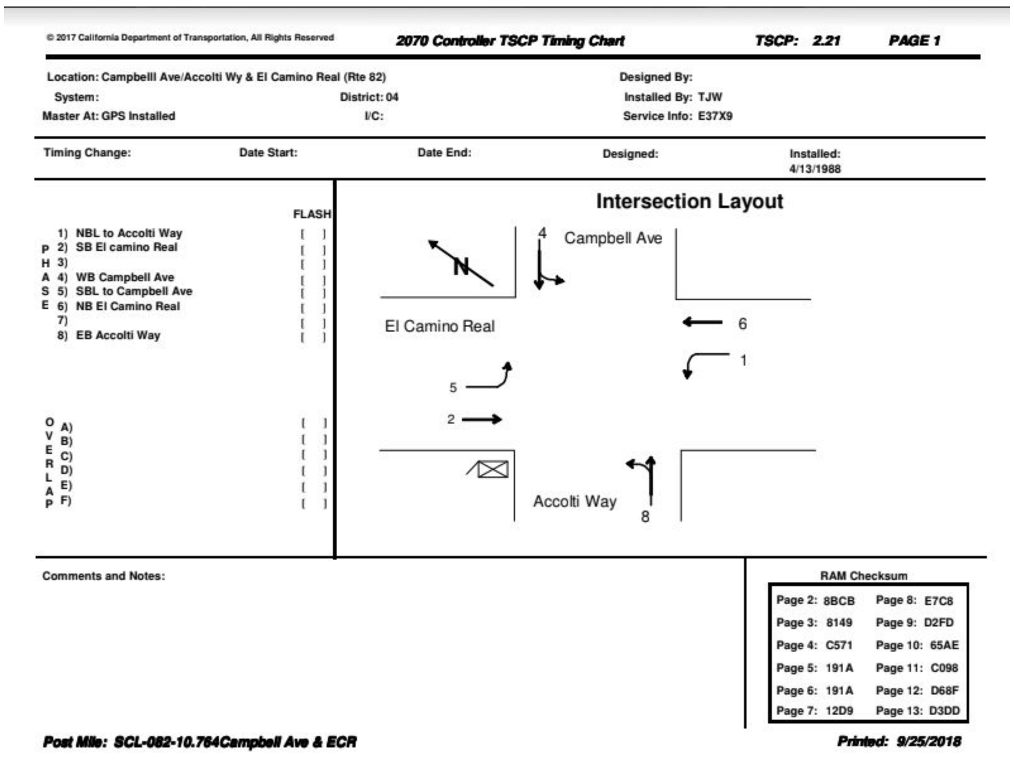
[4] A. Choudhary and IIT Bombay Graduate, “Deep Q-Learning: An Introduction To Deep Reinforcement Learning,” *Analytics Vidhya*, 27-Apr-2020.

[5] T. Patel, “A Q-Learning Approach to Traffic Light Signal Control”

[6] “From The National Traffic Signal Report Card: costs to update signal timing is $3,000 per intersection.,” *ITS*, 2005.

[7] “San Jose TMC Workers Act as Symphony Conductors for City's Roads,” *International Federation of Professional and Technical Engineers*, 01-Nov-2017.

Appendix A


Intersection Layout
N
4
Campbell Ave
El Camino Real
6
1
5
2
Accolti Way
8

© 2017 California Department of Transportation, All Rights Reserved — ***2070 Controller TSCP Timing Chart*** — ***TSCP: 2.21*** — ***PAGE 1***

Location: Campbelll Ave/Accolti Wy & El Camino Real (Rte 82) **Designed By:**
System: **District:** 04 **Installed By:** TJW
Master At: GPS Installed **I/C:** **Service Info:** E37X9

Timing Change: **Date Start:** **Date End:** **Designed:** **Installed:** 4/13/1988

PHASE		FLASH
1)	NBL to Accolti Way	[ ]
2)	SB El camino Real	[ ]
3)		[ ]
4)	WB Campbell Ave	[ ]
5)	SBL to Campbell Ave	[ ]
6)	NB El Camino Real	[ ]
7)		[ ]
8)	EB Accolti Way	[ ]

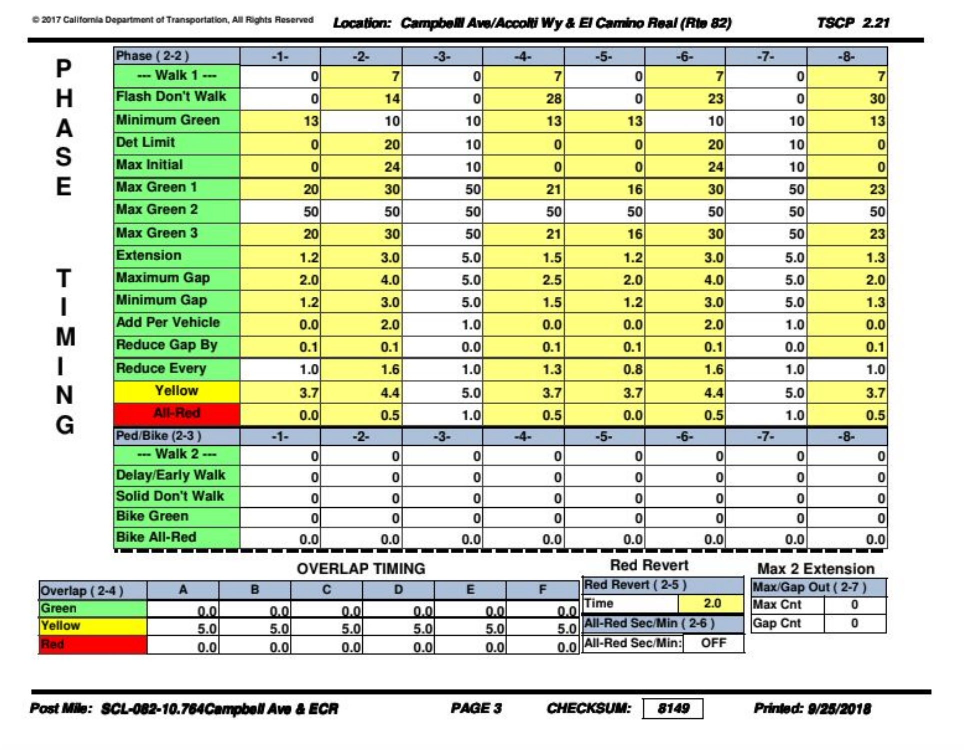
OVERLAP		
A)		[ ]
B)		[ ]
C)		[ ]
D)		[ ]
E)		[ ]
F)		[ ]

Comments and Notes:

RAM Checksum

Page 2:	8BCB	Page 8:	E7C8
Page 3:	8149	Page 9:	D2FD
Page 4:	C571	Page 10:	65AE
Page 5:	191A	Page 11:	C098
Page 6:	191A	Page 12:	D68F
Page 7:	12D9	Page 13:	D3DD

Post Mile: SCL-082-10.764Campbell Ave & ECR ***Printed: 9/25/2018***



© 2017 California Department of Transportation, All Rights Reserved — ***Location: Campbelll Ave/Accolti Wy & El Camino Real (Rte 82)*** — ***TSCP 2.21***

PHASE TIMING

Phase (2-2)	-1-	-2-	-3-	-4-	-5-	-6-	-7-	-8-
--- Walk 1 ---	0	7	0	7	0	7	0	7
Flash Don't Walk	0	14	0	28	0	23	0	30
Minimum Green	13	10	10	13	13	10	10	13
Det Limit	0	20	10	0	0	20	10	0
Max Initial	0	24	10	0	0	24	10	0
Max Green 1	20	30	50	21	16	30	50	23
Max Green 2	50	50	50	50	50	50	50	50
Max Green 3	20	30	50	21	16	30	50	23
Extension	1.2	3.0	5.0	1.5	1.2	3.0	5.0	1.3
Maximum Gap	2.0	4.0	5.0	2.5	2.0	4.0	5.0	2.0
Minimum Gap	1.2	3.0	5.0	1.5	1.2	3.0	5.0	1.3
Add Per Vehicle	0.0	2.0	1.0	0.0	0.0	2.0	1.0	0.0
Reduce Gap By	0.1	0.1	0.0	0.1	0.1	0.1	0.0	0.1
Reduce Every	1.0	1.6	1.0	1.3	0.8	1.6	1.0	1.0
Yellow	3.7	4.4	5.0	3.7	3.7	4.4	5.0	3.7
All-Red	0.0	0.5	1.0	0.5	0.0	0.5	1.0	0.5
Ped/Bike (2-3)	**-1-**	**-2-**	**-3-**	**-4-**	**-5-**	**-6-**	**-7-**	**-8-**
--- Walk 2 ---	0	0	0	0	0	0	0	0
Delay/Early Walk	0	0	0	0	0	0	0	0
Solid Don't Walk	0	0	0	0	0	0	0	0
Bike Green	0	0	0	0	0	0	0	0
Bike All-Red	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

OVERLAP TIMING

Overlap (2-4)	A	B	C	D	E	F
Green	0.0	0.0	0.0	0.0	0.0	0.0
Yellow	5.0	5.0	5.0	5.0	5.0	5.0
Red	0.0	0.0	0.0	0.0	0.0	0.0

Red Revert

Red Revert (2-5)	
Time	2.0
All-Red Sec/Min (2-6)	
All-Red Sec/Min:	OFF

Max 2 Extension

Max/Gap Out (2-7)	
Max Cnt	0
Gap Cnt	0

Post Mile: SCL-082-10.764Campbell Ave & ECR ***PAGE 3*** ***CHECKSUM: 8149*** ***Printed: 9/25/2018***

Intersection Layout and Signal Design for El Camino Real & Campbell

Appendix B

Webster’s Cycle Model

$$C = \frac{1.5L + 5}{1.0 - Y}$$

where:

C	= optimum cycle length, (s)
Y	= critical lane volume divided by the saturation flow, summed over the phases
L	= lost time per cycle, (s).

Equation provided by Federal Highway Administration[1]

Highway Capacity Manual (HCM) Cycle Model

$$C_{des} = \frac{LT}{1 - \dfrac{Vc}{1615PHF(v/c)}}$$

Where LT = total lost time per cycle

v/c = volume to capacity ratio desired

PHF = peak hour factor

V_c = critical volume

Appendix C

```
class DQNAgent: #DNN to approximate Q learning state action algorithm
        def __init__(self):
        self.gamma = 0.95  # discount rate
        self.epsilon = 0.1 # exploration rate
        self.learning_rate = 0.0002 #0.0002
        self.memory = deque(maxlen=250)
        self.model = self._build_model()
        self.action_size = 5

        def _build_model(self):
        # Neural Net for Deep-Q learning Model
        input_1 = Input(shape=(12, 12, 1))
        x1 = Conv2D(16, (4, 4), strides=(2, 2), activation='relu')(input_1)
        x1 = Conv2D(32, (2, 2), strides=(1, 1), activation='relu')(x1)
        x1 = Flatten()(x1)

        input_2 = Input(shape=(12, 12, 1))
        x2 = Conv2D(16, (4, 4), strides=(2, 2), activation='relu')(input_2)
        x2 = Conv2D(32, (2, 2), strides=(1, 1), activation='relu')(x2)
        x2 = Flatten()(x2)

        input_3 = Input(shape=(5, 1))
        x3 = Flatten()(input_3)

        x = keras.layers.concatenate([x1, x2, x3])
        x = Dense(128, activation='relu')(x)
        x = Dense(64, activation='relu')(x)
        x = Dense(5, activation='linear')(x)

        model = Model(inputs=[input_1, input_2, input_3], outputs=[x])
        model.compile(optimizer=keras.optimizers.RMSprop(
        lr=self.learning_rate), loss='mse')

        return model
```

```
#example of state action selection algorithm
        if(action == 3 and light[0][0][0] == 1): #transition from state 0-1-2-3
        # Transition Phase
        for i in range(ty):
        stepz += 1
        traci.trafficlight.setPhase('0', 1)
        wait_time()
        traci.simulationStep()
        for i in range(tg):
        stepz += 1
        traci.trafficlight.setPhase('0', 2)
        wait_time()
        traci.simulationStep()
        # Action Execution
        reward1 = traci.edge.getLastStepVehicleNumber('2si')
        reward2 = traci.edge.getLastStepHaltingNumber('3si') + traci.edge.getLastStepHaltingNumber('4si') +
traci.edge.getLastStepHaltingNumber('1si')
        for i in range(tg):
        stepz += 1
        traci.trafficlight.setPhase('0', 3)
        reward1 = traci.edge.getLastStepVehicleNumber('2si')
        reward2 = traci.edge.getLastStepHaltingNumber('3si') + traci.edge.getLastStepHaltingNumber('4si') +
traci.edge.getLastStepHaltingNumber('1si')
        wait_time()
        traci.simulationStep()
```