An Integratable V-X Communication based Conventional Vehicle Fuel Optimization model for distance based Ecological Driving Scheme

by

Prudhvi Raj Rongali

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering (Computer Engineering) in the University of Michigan-Dearborn 2018

Master's Thesis Committee:

Assistant Professor Wencong Su, Chair
Assistant Professor Mengqi Wang
Assistant Professor Riadul Islam

ACKNOWLEDGEMENTS

First and foremost I wish to thank my advisor, Dr. Wencong Su, Assistant Professor in the Department of Electrical and Computer Engineering. Whose guidance, support and encouragement from the initial to the final level of completion of my Thesis work.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

ABSTRACT

This thesis paper proposes An Integratable V-X Communication-based Conventional Vehicle Fuel Optimization model for real-time traffic conditions. Before departure, the speed profile for an entire route is optimized using signal phase and timing $(SPAT)$ information and location of traffic lights to provide smooth transitions at traffic signal intersections. In this study, we are going to develop ”nonstop” optimal speed model that can be integrated to existing distance based eco-driving schemes. The initial simulation is done using MATLAB to evaluate optimal speed, fuel economy, the travel time of the ”nonstop” model and the results are compared with the optimization results from distance based eco-driving scheme which uses an estimation of distribution algorithm (EDA). Further integration compatibility of ”nonstop” model with the distance based eco-driving scheme is analyzed.

CHAPTER I
Introduction

Energy consumption optimization and pollution reduction are increasingly becoming critical. For the past decade the number of vehicles on road are increasing exponentially. This lead to serious debate in developing vehicle optimization techniques to cut the emissions. The European Union passed the legislation to cut the emissions by 20 percent towards the end of 2020 [1]. The figure 1.1 describes the projections and targets in the EU by 2020 and beyond.

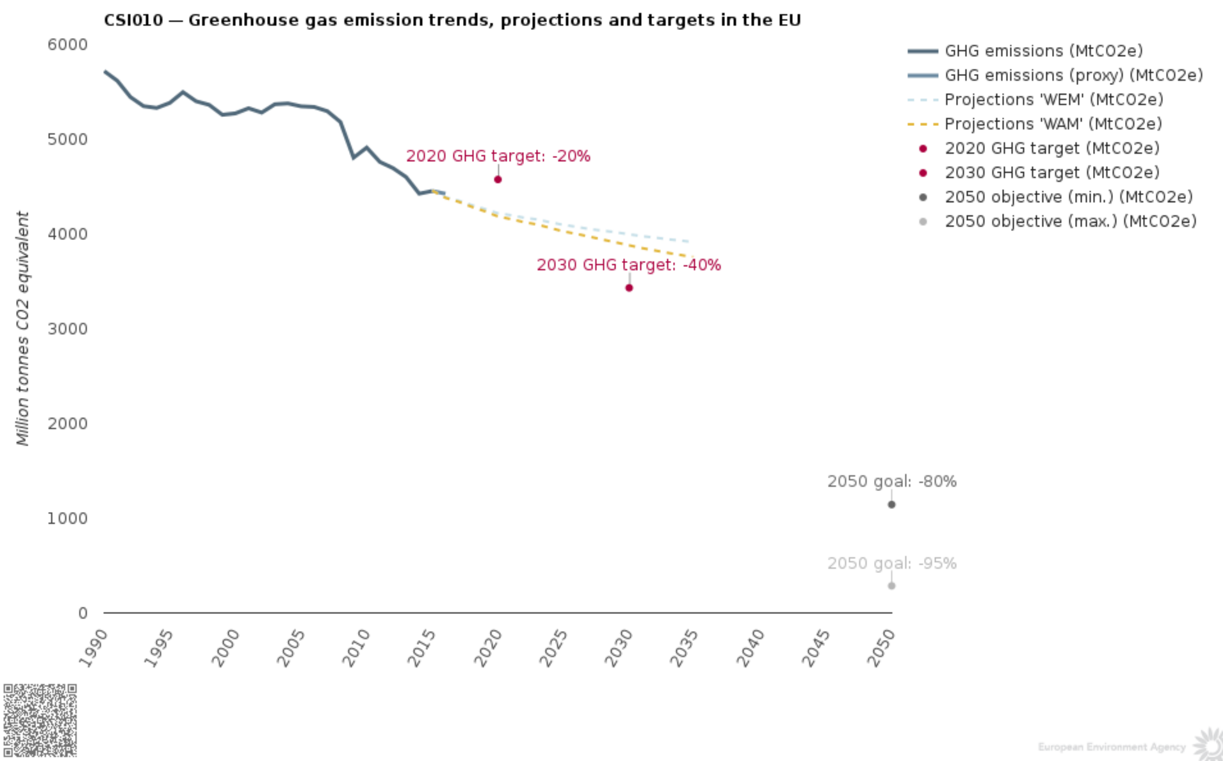


Figure 1.1: EU green house gases emission trends, projections[1].

This similar trend can be seen around the world. The energy efficiency of a vehicle can be affected by traffic conditions and driving behavior. The behavior of a driver that minimizes fuel consumption is often referred to as eco-driving. The Eco driving systems helps the vehicles to operate energy efficiently. A distance -based Ecological driving scheme [2] proposes Two-stage Ecological Driving scheme by utilizing estimation of distribution algorithm. This approach optimizes the speed profile without compromising on the fuel efficiency by considering the drivetrain parameters, speed limits of the road, distance from the preceding vehicle. This approach is further discussed in detail in $chapter4$. There are other research works that shows that by avoiding the complete full stop at traffic intersections will reduce the fuel consumption by 12 to 14 percentage, which can be achieved through advising cruising velocities to the drivers [3]. The figure 1.2 shows the fuel savings by crusing rather than idling at traffic intersections.

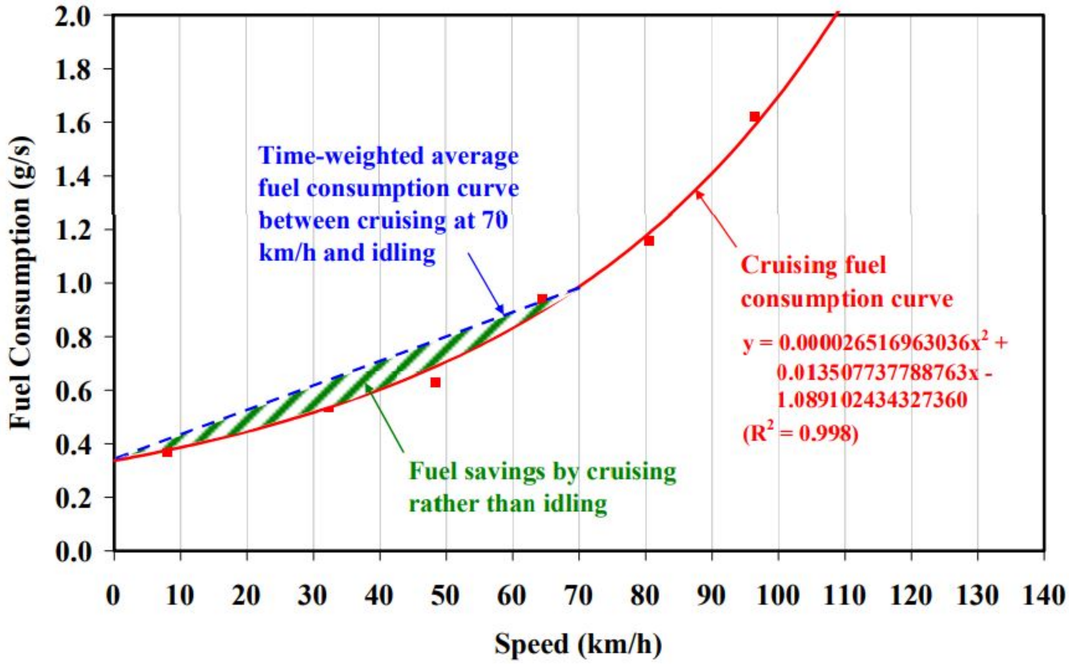


Figure 1.2: Fuel consumption vs. cruise speed[3].

Extensive research is being carried to develop adaptive traffic control systems such as SCOOT [4] SCATS and TUC [5] has been implemented around the world to reduce the traffic congestion and emissions. The SCOOT systems continously monitors the traffic flow in the whole traffic network and adjusts the signal timings to reduce the delays and make the traffic flow smoother. The figure 1.3 shows the signal co-ordination with a fixed time plan.

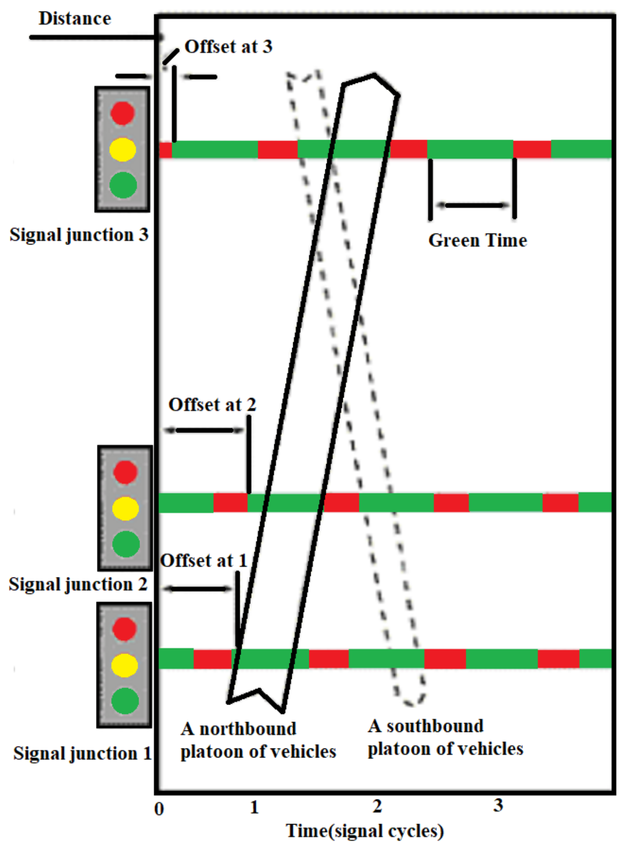


Figure 1.3: An idealized time-distance diagram showing signal co-ordination with a fixed time plan[6].

Due to the state of art in wireless communication and reducing prices in the electronics that helps to build V2I communication systems that helps to communicate the location and timing of traffic signals with advanced driving assistance systems (ADAS). This thesis paper proposes "*nonstop*" feature improvement model for Two-stage Ecological Driving scheme by utilizing the real time signal phasing and timing information(SPAT) and traffic signal position information, average speed of vehicles. This approach utilizes the existing vehicle and fuel models from Two-stage Ecological Driving scheme to not lose the generality with existing model. Further this feature generates optimal speeds in a way that the vehicle can cross the traffic intersections without stopping in urban environment. Typical urban environment shown in figure 1.4.

The contributions of this thesis paper are mainly as in the following:

1) By using the SPAT information, traffic flow using V2I communication figure 1.5 the speed tra-

Figure 1.4: Example of urban traffic environment[9].

jectory can be optimized by narrowing down the search space for green phase windows[7], We further improve this model with an Integrated matrix index mapping module to create adjacency matrix for graph with multiple inward and outward edges;

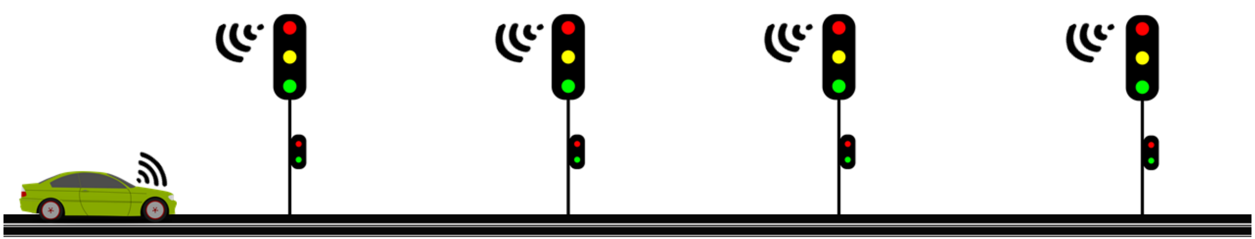


Figure 1.5: V2I communication in urban traffic environment.

2) Overall less computational time for online applications;

3) Integration compatibility with Two-stage Ecological Driving scheme (long term optimization [2]);

4) An optimized approach which is easily applicable for EV and HEV considering the vehicle model and energy cost functions are given;

5)This model can also potentially run in congestion with distance based local optimization [2].

1.1 Context

There is active research going on to study the effect of driving style on fuel consumption. Some of the common terms are defined here.

1.1.1 Definitions

Driving Style : Driver style refers to the level of aggressiveness. The driving style can be differentiated based on the acceleration, velocity values for a period of time.

Fuel Economy : It's expressed a miles per gallon. Also it is a relation between the fuel consumed for a particular speed trajectory.

Fuel Consumption : Fuel consumption is an inverse of fuel economy and it's expressed as miles per gallon.

Fuel use : It's the total amount of fuel consumed in a particular journey from source to destination.

Efficiency : Efficiency is the ratio of power out to power in.

1.2 Eco-driving

Eco driving is a driving style that uses less fuel. This can be achieved by driving at economical speed, without sudden accelerations and idling due to traffic congestion. The eco driving has more potential to improve the fuel economy in urban environment compared to the highways; Due to the fact that there is less chances to come across traffic stops or idling due to the traffic congestion.

CHAPTER II

Vehicle and SPAT model

This work objective is minimization of energy consumed by vehicles. The optimization is performed by considering vehicle model and traffic signal information[7] and the speed profile is generated with constraints on the traffic crossing times only occur during the signal is in green phase along the distance horizon. In this "nonstop" model we consider that we have full knowledge of traffic light timings and driving horizons communicated via V2I infrastructure.

2.1 Vehicle Model

The vehicle and fuel rate functions are derived from the Two-stage Ecological Driving scheme [2]. The relation between fuel rate , torque and engine speed in [8] are modeled based on the Willan's line approximation using autonomie. The figures 2.1 and 2.2 are the representation of the above mentioned relationship in piece-wise linear form.

In figure 2.2 for any fixed value of torque the fuel rate model $\dot{m}_f$ can be represented as linear function of engine speed ω and torque T_e as shown below.

$$\dot{m}_f = f\left(T_c, \omega\right) = \left(\beta_1 \omega + \beta_2\right) T_e + \gamma_1 \omega + \gamma_2 \tag{2.1a}$$

The authors also evaluated the fuel rate model in figure 2.2 is acceptable by comparing it with fuel map generated by Autonomie, The solid lines and dashed lines represent the data from autonomie and fuel rate model. The fuel map from autonomie is show in figure 2.3

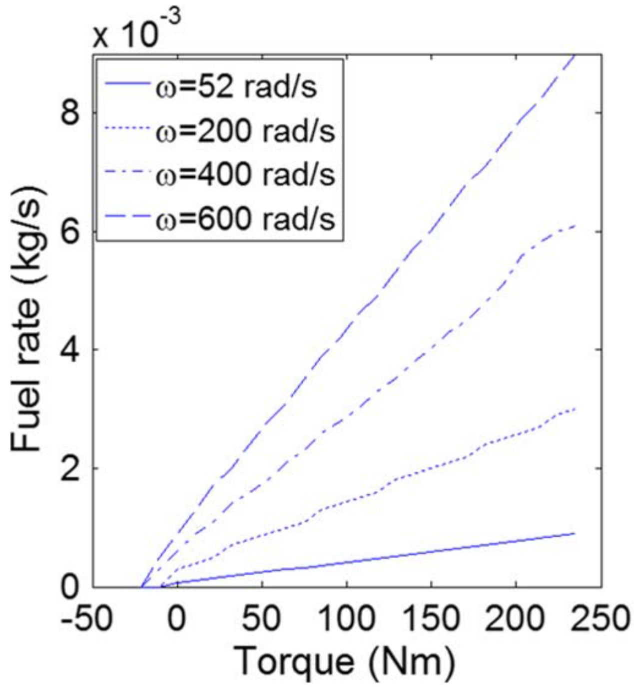


Figure 2.1: Fuel rate vs Torque[8].

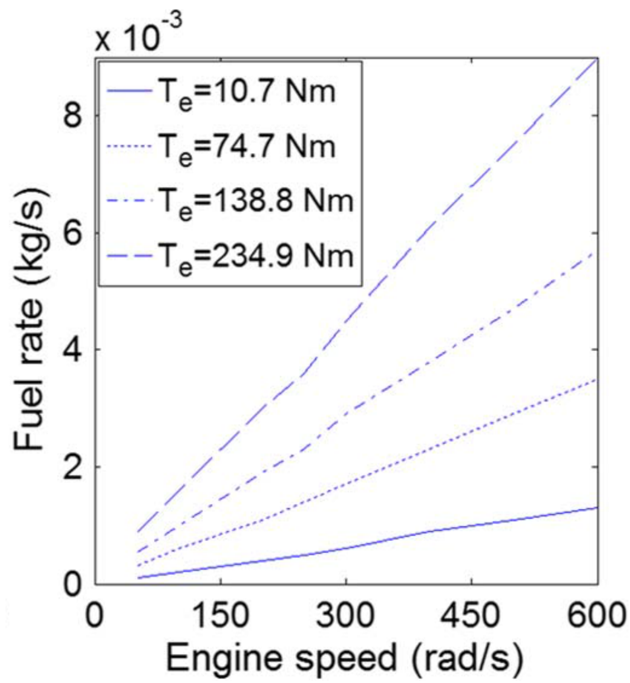


Figure 2.2: Fuel rate vs Engine speed[8].

The above fuel rate model 2.1a is used as a weight function in following chapter 3 to determine the optimal speed profile. As the speed profile is used as a feedback to the driver it must be represented in terms vehicle velocity v (m/s). So, the engine speed ω can be represented as

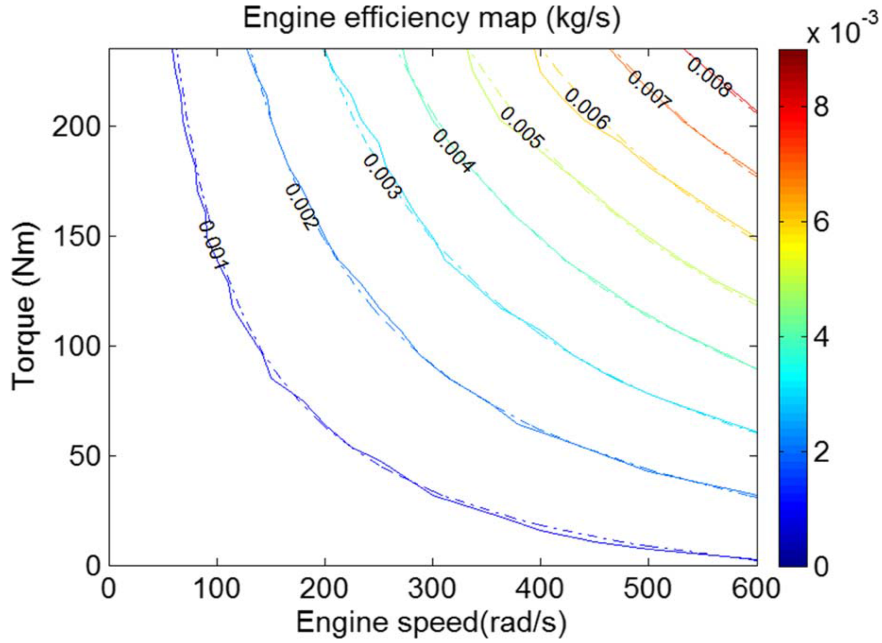


Figure 2.3: Fuel map[8].

$$\omega = \frac{f_r g_r(n) * v}{r_w} \tag{2.1b}$$

where f_r is a final drive ratio, $g_r(n)$ is a gear ratio for a given gear number n, and r_w is a wheel radius (m). By substituting Equation 2.1b into fuel rate function 2.1a gives us following equation:

$$\dot{m}_f = \left(\beta_1 \frac{f_r g_r(n)}{r_w} v + \beta_2\right) T_e + \gamma_1 \frac{f_r g_r(n)}{r_w} v + \gamma_2 \tag{2.1c}$$

where v is vehicle speed β_1,β_2,γ_1,γ_2 are constants.

The constant values are:

$$\beta_1 = 5.646 \times 10^{-8}$$

$$\beta_2 = 4.751 \times 10^{-7}$$

$$\gamma_1 = 1.625 \times 10^{-6}$$

$$\gamma_2 = -5.968 \times 10^{-5}$$

The model parameters as described in Table 2.1:

Parameters	Values	Unit
m	1607	kg
fr	4.438	-
Nfr	0.97	-
rw	0.30115	m
gr(n)	2.563,1.552,1.022,0.727,0.52,0.97	-

Table 2.1: Vehicle parameters[8].

We use a deterministic approach to compute the signal phasing and timing (SPAT) information. It is assumed that the time offsets are communicated using V2I infrastructure. This model can represented mathematically in the following section.

2.2 SPAT Model

$$s_i(t) = \begin{cases} 1, & T_i^0 + kT_i < t \leq T_i^0 + kT_i + T_i^G \\ 0, & T_i^0 + kT_i + T_i^G < t \leq T_i^0 + (k+1)T_i \end{cases} \tag{2.1d}$$

The above expression 2.1d is derived from the [7], used to determine the state $s_i(t)$ of the ith traffic light at any point of journey. T_i^0 is the initial phase time, T_i is the total phase duration(green + red), T_i^G is duration of green phase, k is number of phase cycles. If the state equals '1' it means the $i$the traffic light is in green state at time t otherwise; if it's equal to '0' the traffic light is in red state. For the driver's safety consideration the yellow light phase duration is combined with Red phase duration.

In this thesis paper we are considering six traffic lights and the above mathematical model 2.1d can be used to determine the signal status of any of the six traffic lights on it's path. The following figures 2.4,2.5, 2.6, 2.7, 2.8,2.9 represents the signal phase changes for six traffic lights over a period of 400 seconds.

Traffic light 1 signal status:

$T_i^0 = 20\ sec$

$T_i = 15\ sec$

$T_i^G = 15\ sec$

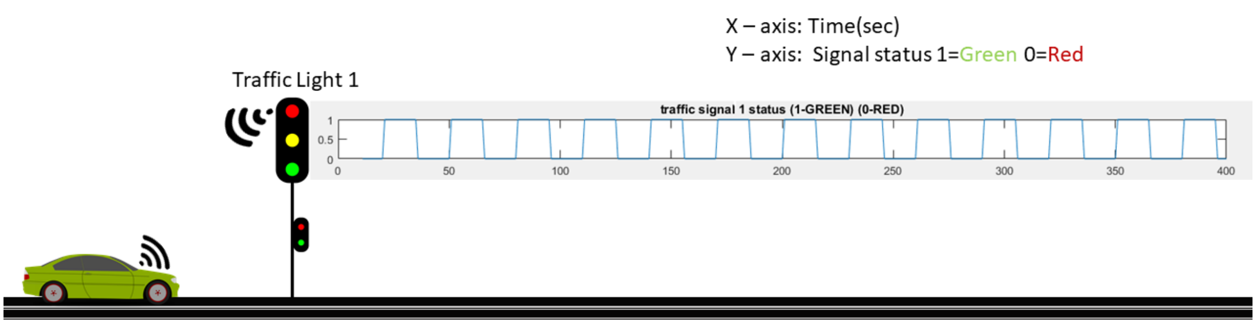


Figure 2.4: Signal status for Traffic light 1.

Traffic light 2 signal status:

$T_i^0 = 5\ sec$

$T_i = 8\ sec$

$T_i^G = 8\ sec$

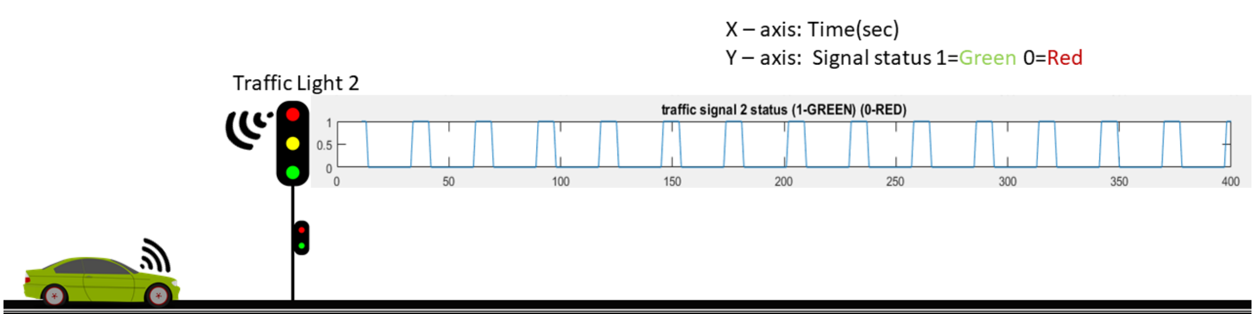


Figure 2.5: Signal status for Traffic light 2.

Traffic light 3 signal status:

$T_i^0 = 10\ sec$

$T_i = 25\ sec$

$T_i^G = 25\ sec$

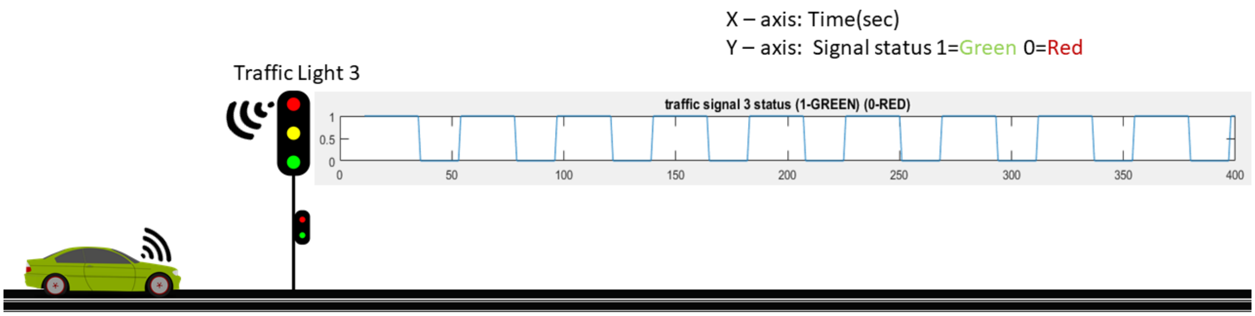


Figure 2.6: Signal status for Traffic light 3.

Traffic light 4 signal status:

$T_i^0 = 10\ sec$

$T_i = 15\ sec$

$T_i^G = 15\ sec$

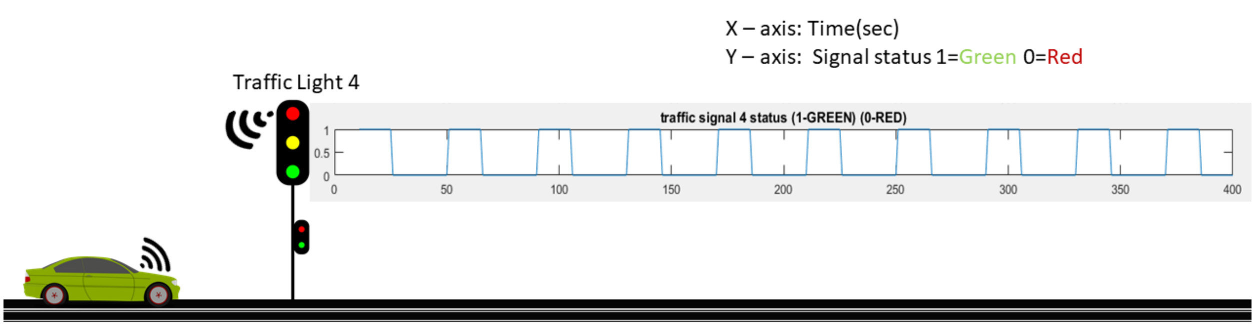


Figure 2.7: Signal status for Traffic light 4.

Traffic light 5 signal status:

$T_i^0 = 20\ sec$

$T_i = 15\ sec$

$T_i^G = 15\ sec$

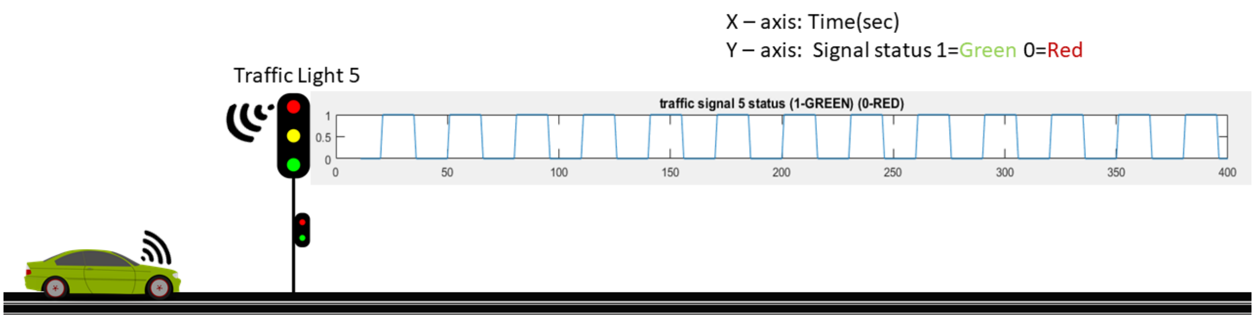


Figure 2.8: Signal status for Traffic light 5.

Traffic light 6 signal status:

$T_i^0 = 10\ sec$

$T_i = 20\ sec$

$T_i^G = 20\ sec$

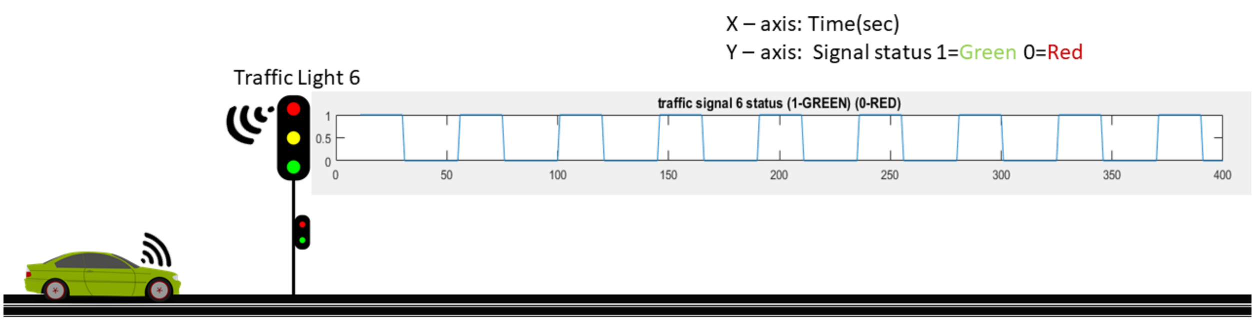


Figure 2.9: Signal status for Traffic light 6.

When the vehicle is approaching the 6th traffic light with in 200 seconds the vehicle can have a very smooth transition at the traffic intersection without stopping as shown in the figure 2.10.

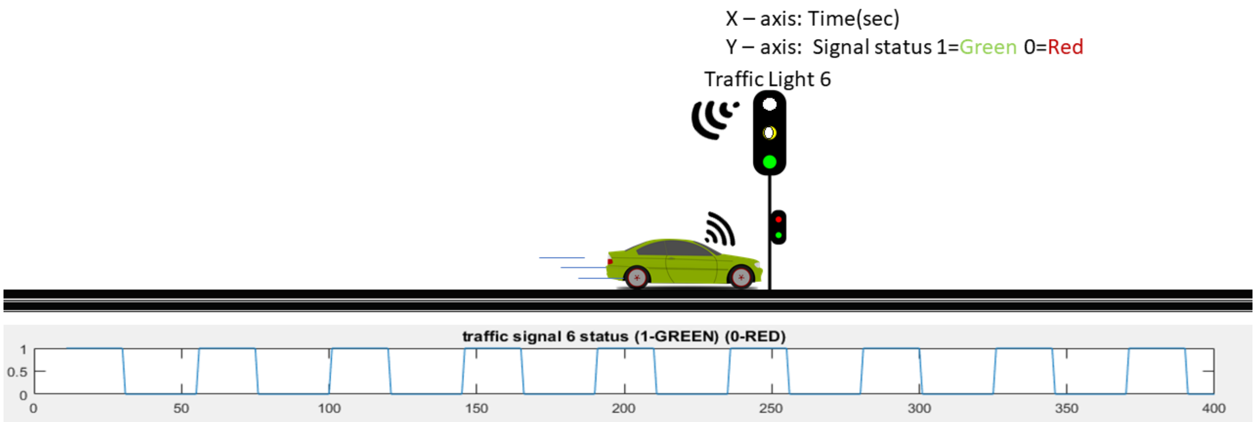


Figure 2.10: Vehicle passing through traffic intersection during Green phase.

CHAPTER III

Optimization problem

The optimization problem is to find optimal speed profile using Dijkstra algorithm among all the available crossing speeds at each traffic intersection. The weighted matrix consists of the fuel energy consumed in each path.

Algorithm 1 Search of Possible Intersection Crossing Time Windows

$t_{\text{fast}} = \frac{d_i - d_{i-1}}{V_{\max}} + t_{i-1}^{\min}$;
$j = 1$;
$t_{i,j}^{\max} =$ next instance after t_{fast}, when light switches to red;
$t_{i,j}^{\min} = \max(t_{\text{fast}}, t_{i,j}^{\max} - T_i^G)$;
$t_{\text{slow}} = \frac{d_i - d_{i-1}}{V_{\min}} + t_{i-1}^{\max}$;
while $(t_{i,j}^{\min} + T_i < t_{\text{slow}})$ **do**
  $j = j + 1$;
  $t_{i,j}^{\min} = t_{i,j-1}^{\min} + T_i$;
  $t_{i,j}^{\max} = \min(t_{i,j}^{\min} + T_i^G, t_{\text{slow}})$;
end while

Figure 3.1: Crossing window algorithm[7].

The following sections describe the algorithm used to create the weighted adjacency matrix using the feasible crossing windows at each intersection. The crossing window search algorithm adapted from [7] uses the speed profile calculation based on basic discrete time integration and basic vehicle kinematic equations. Figure 3.1 shows the authors implementation of cross window search algorithm. The authors algorithms determines all the possible crossing windows at each intersection but doesn't generate the adjacency matrix; which can be used as an input for various optimization algorithms. Here we extend that algorithm further to generate adjacency matrix in the same code routine, which can be populated with either crossing times or fuel consumption values as weights to generate the rough estimation of the optimal speed. Further, the authors used Sequential Quadratic Programming (SQP) method to solve the simplified and the original optimization problem proposed in paper[7]. In this paper we use Dijkstra algorithm to find the optimal path. The Dijkstra algorithm is feasible for this approach compared to Floyd-Warshall algorithm because the weights generated are always positive.

3.1 Cross window search and Adjacency matrix creation Algorithm

The first step is to find all possible crossing windows during the green phase at each traffic signal intersection. Once the crossing windows are determined we will incorporate matrix indexing code routine to convert a regular matrix in to adjacency matrix.

Psedo code 1: Cross window search and Adjacency matrix creation Algorithm

---

1: **for** i=1 : number of traffic lights

2 : $V_{max} = min(max\ speed\ limit, V_{flow})$

3: if i=1; Absolute distance=d_i **else equals** $d_i - d_{i-1}$

4 : $if\ i = 1; t_{fast} = \frac{Absolute\ distance}{V_{max}}$ **else equals** $t_{\text{fast}} = \frac{Absolute\ distance}{V_{max}} + t_{i-1}^{\min}$

5 : $find\ the\ i^{th}\ signal\ status\ at\ t_{fast}$

6 : **while** : $if\ status\ is\ red\ increment\ the\ search\ variable\ till\ green\ phase\ is\ found$

7 : **end while**

8 : $t_{i,j}^{\max} = next\ green\ phase$

$9: t_{i,j}^{\min} = \max\left(t_{\text{ fast}}, t_{i,j}^{\max} - T_i^G\right)$

$10: if\ i = 1; t_{\text{slow}} = \frac{d_i - d_{i-1}}{V_{\min}}$**else equals**$t_{\text{slow}} = \frac{d_i - d_{i-1}}{V_{\min}} + t_{i-1}^{\max}$

$11:$ while $\left(t_{i,j}^{\text{tmin}} + T_i < t_{\text{ slow}}\right)$ do

$12: j = j + 1$

$13: t_{i,j}^{\min} = t_{i,j-1}^{\min} + T_i$

$14: t_{i,j}^{\max} = \min\left(t_{i,j}^{\min} + T_i^G, t_{\text{slow}}\right)$

15 : **end while**

16 : concatenate $t_{i,j}^{\min}$ and $t_{i,j}^{\max}$

17 : Sort the above array in ascending order colounm wise and store in $t_{combine}$

18 : Nnzvr2 represents number of non zero values in $(i-1)_{th}$ row

19 : Nnzvr2 values we determine for each node if the egde is going inwards and outwards

20 : For each iteration adjacency matrix's for speed(speedadj) and fuel(fuelwtadj) consumption are populated.

21 : For fuelwtadj matrix negative weights are made zero

22 : Both the adjacency matrix's are converted in to square matrix

23 : **end while**

In the above **Psedo code 1** steps 1 through 14 minimizes the search space of available cross windows during green phases by identifying only feasible windows that will allow the driver to reach the destination without stopping at any traffic intersection by compiling to the speed limits and traffic flow. For each iteration we first compute the V_{max}. Which is the minimum value among the maximum speed and traffic flow speed. After that we calculate the t_{fast} to check whether the vehicle can reach the i_{th} traffic intersection before the traffic light turns to red. If the computed t_{fast} wouldn't help the vehicle to reach during green crossing time; we will find the next possible green window and assign that time to $t_{i,j}^{\max}$. If the computed $t_{i,j}^{\text{tmin}} + T_i < t_{\text{ slow}}$ it means that there are still feasible crossing windows available. These new feasible windows are populated and stored in $t_{i,j}^{\max}$ and $t_{i,j}^{\min}$.

3.2 Adjacency matrix and graph reduction

The Steps 16 to 22 will help to map the values in $t_{i,j}^{\max}$,$t_{i,j}^{\min}$ into an adjacency matrix. The vehicle can cross the intersection in green phase as long as the vehicle can reach the intersection in between [$t_{i,j}^{\max}$, $t_{i,j}^{\min}$]. This will enable us to several nodes in this time range. Considering the complicity will increase at the rate of $\theta(log(n))$ with increase in number of nodes for each crossing window one can consider midpoint of [$t_{i,j}^{\max}$, $t_{i,j}^{\min}$] as a single node.

In this approach we are mapping the 2 nodes on the each end of the crossing windows. As in the step 16 we concatenate and sort both min and max crossing times in ascending order and resultant matrix cannot be used for optimization; this matrix need to be transformed in to adjacency matrix where nodes and edges are defined properly. The steps 18,19 used for mapping the data in to an adjacency matrix and these code routines are very specific to this problem. While building the adjacency matrix the weights are calculated using following using the 2.1c as weight function:

$$\dot{m}_{f,total} = (\left(\beta_1 \frac{f_r g_r(n)}{r_w} v + \beta_2\right) T_e + \gamma_1 \frac{f_r g_r(n)}{r_w} v + \gamma_2) * \delta_t \tag{3.1}$$

$$\delta_t = \frac{d_i - d_{i-1} - 25}{v} \tag{3.2}$$

Where δ_t time duration of journey between two intersections and v is the vehicle speed between i_{th} and $(i-1)_{th}$ traffic signals. Also, we consider the transition between two intersection occur smoothly with in 25mt distance. Apart from this the speed will be maintained at constant value for the distance of $d_i - d_{i-1} - 25$. Further substitute 3.2 in 3.1 we get.

$$\dot{m}_{f,total} = (\left(\beta_1 \frac{f_r g_r(n)}{r_w} v + \beta_2\right) T_e + \gamma_1 \frac{f_r g_r(n)}{r_w} v + \gamma_2) * (\frac{d_i - d_{i-1} - 25}{v}) \tag{3.3}$$

The transition phase fuel consumption parameters are computed after the optimal speed profile is determined.

3.3 Optimal path

The wights of adjacency matrix are always positive in this application as the fuel rate is positive variable and cannot go negative. For non negative weights Dijkstra algorithm is best suitable optimization algorithm. We are using the Dijkstra in this application because of it's simple approach and faster computing time considering our real-time use in driving scenarios.

Before apply the the Dijkstra algorithm. We need to transform the adjacency matrix into square matrix. After the conversion the algorithm can be applied. In this work we are using the Dijkstra algorithm from matlab toolbox[mlab]. The algorithm provides route with low fuel consumption. This information is used to create the optimal speed profile. The application of Dijkstra algorithm and simulation results are further discussed in the following chapter 4.

For example for the following parameters; if we run psedo code 1 with the following parameters:

$$V_{min} = 14\, mts/sec$$

$$V_{speed_limit} = 35\, mts/sec$$

$$V_{flow} = [16, 25, 25, 21, 16, 24]\, mts/sec$$

$$d_i = [800, 1500, 2200, 3000, 3500, 4500]\, mts$$

The results for cross windows regions are as given in the for each traffic light:

Traffic light 1:

$$t_{min} = 51\, sec$$

$$t_{max} = 66\, sec$$

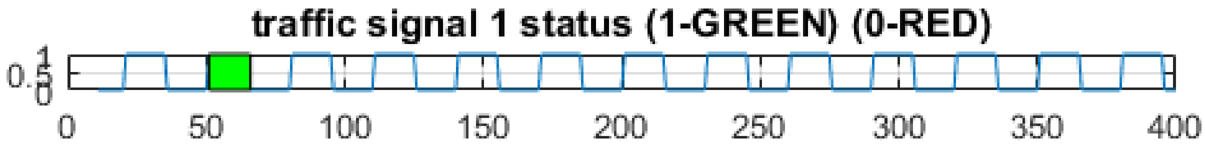


Figure 3.2: Crossing window for Traffic light 1. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 2:

$$t_{min} = 90 \; sec$$

$$t_{max} = 98 \; sec$$

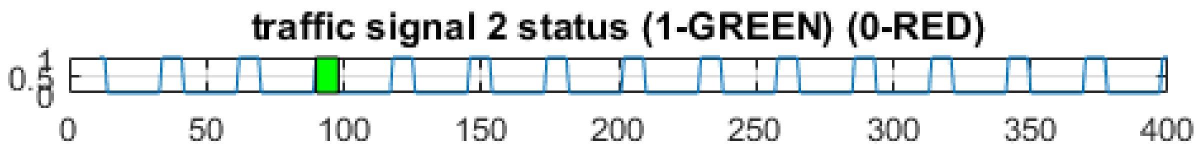


Figure 3.3: Crossing window for Traffic light 2. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 3:

$$t_{min} = 118 \; sec$$

$$t_{max} = 122 \; sec$$

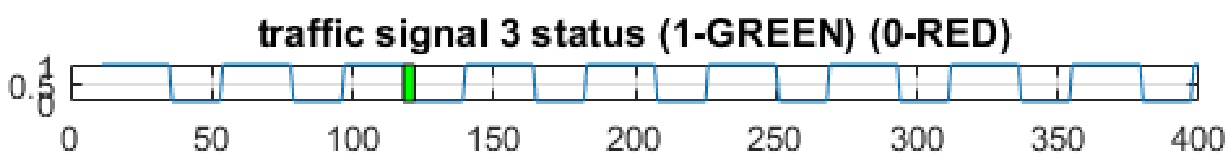


Figure 3.4: Crossing window for Traffic light 3. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 4:

$$t_{min} = 170.09 \; sec$$

$$t_{max} = 185.09 \; sec$$

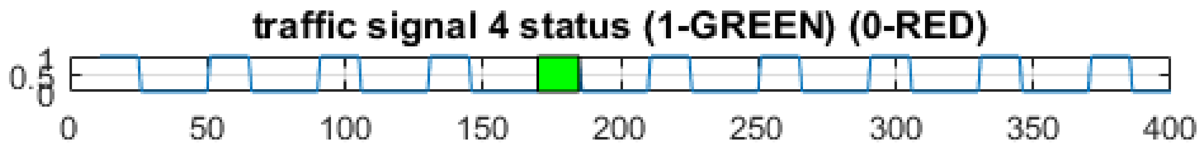


Figure 3.5: Crossing window for Traffic light 4. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 5:

$$t_{min} = 201.34\ sec$$

$$t_{max} = 215.34\ sec$$

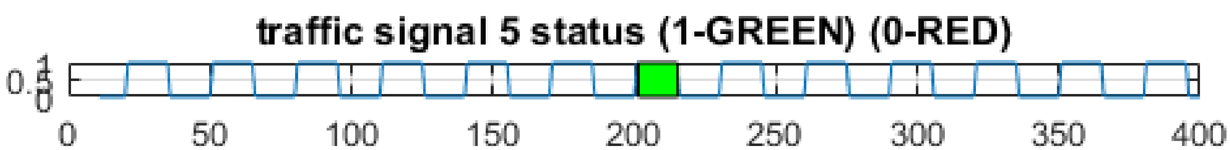


Figure 3.6: Crossing window for Traffic light 5. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 6:

$$t_{min} = 243.01\ sec$$

$$t_{max} = 255.01\ sec$$

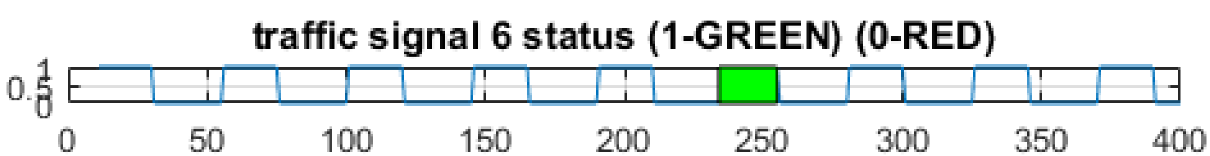


Figure 3.7: Crossing window for Traffic light 6. x-axis: time(sec), y-axis:signal status(1 or 0).

The t_{min} represents figure 3.8 the earliest time vehicle cross the traffic intersection without stopping and t_{max} represents figure 3.9 the final time the vehicle has to reach the traffic light before traffic light turn red.

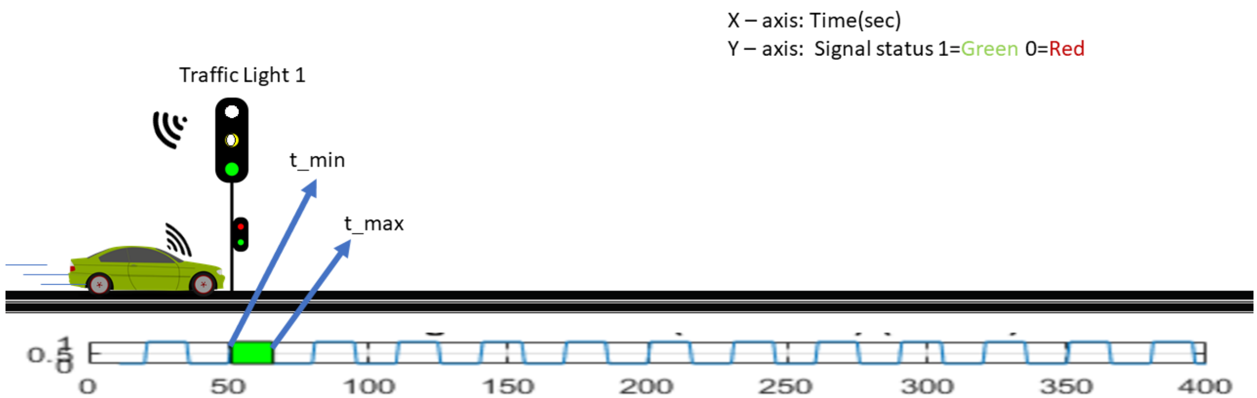


Figure 3.8: t_{min} crossing point for Traffic light 1.

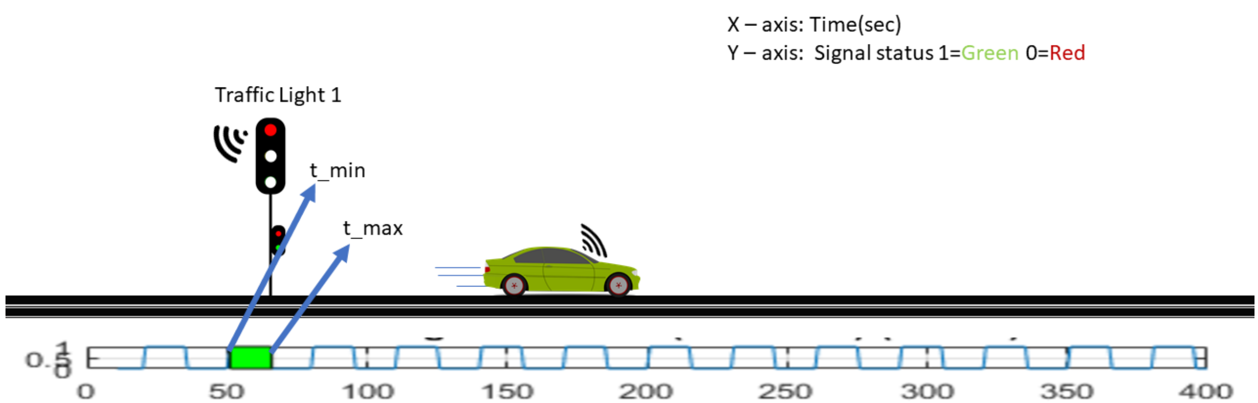


Figure 3.9: t_{max} crossing point for Traffic light 1.

In order for the vehicle to reach the destination without stopping; The vehicle has to cross the intersection during the green windows figure 3.10 . By using the optimal path approach the vehicle not only crosses the green windows but also it can be optimized to find the best fuel economical path.

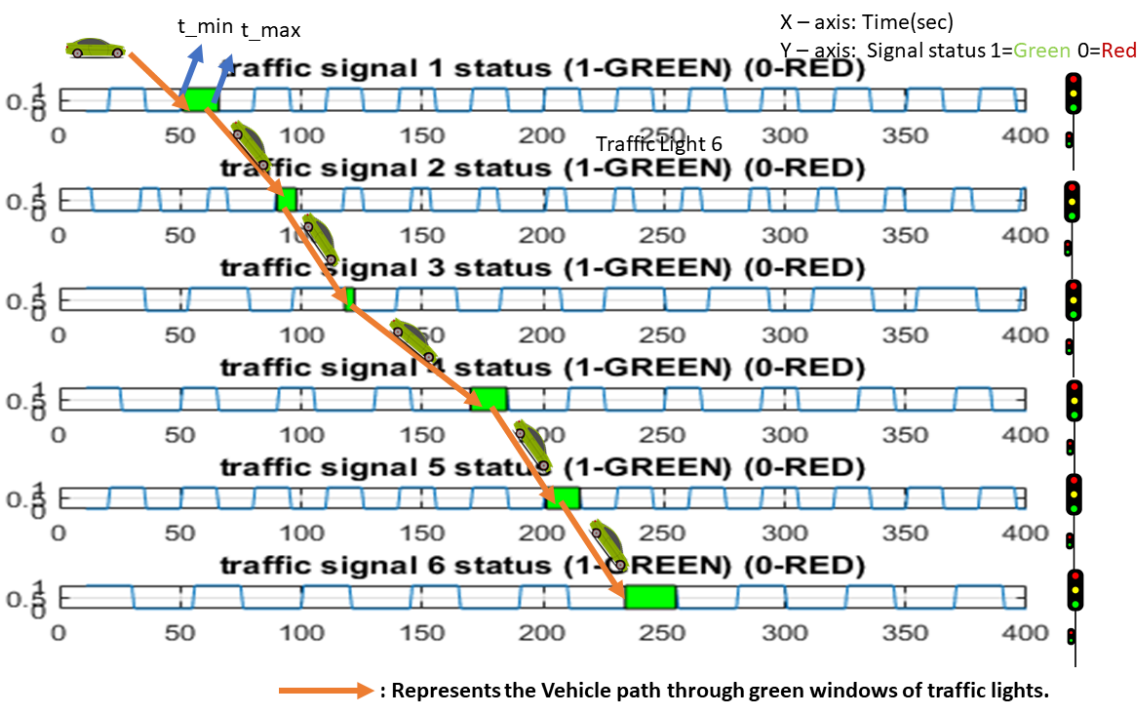


Figure 3.10: Vehicle trajectory through green crossing windows.

For the optimization using Dijkstra algorithm each t_{max} and t_{min} values for 6 traffic lights are considered as individual nodes as shown in figure 3.11.

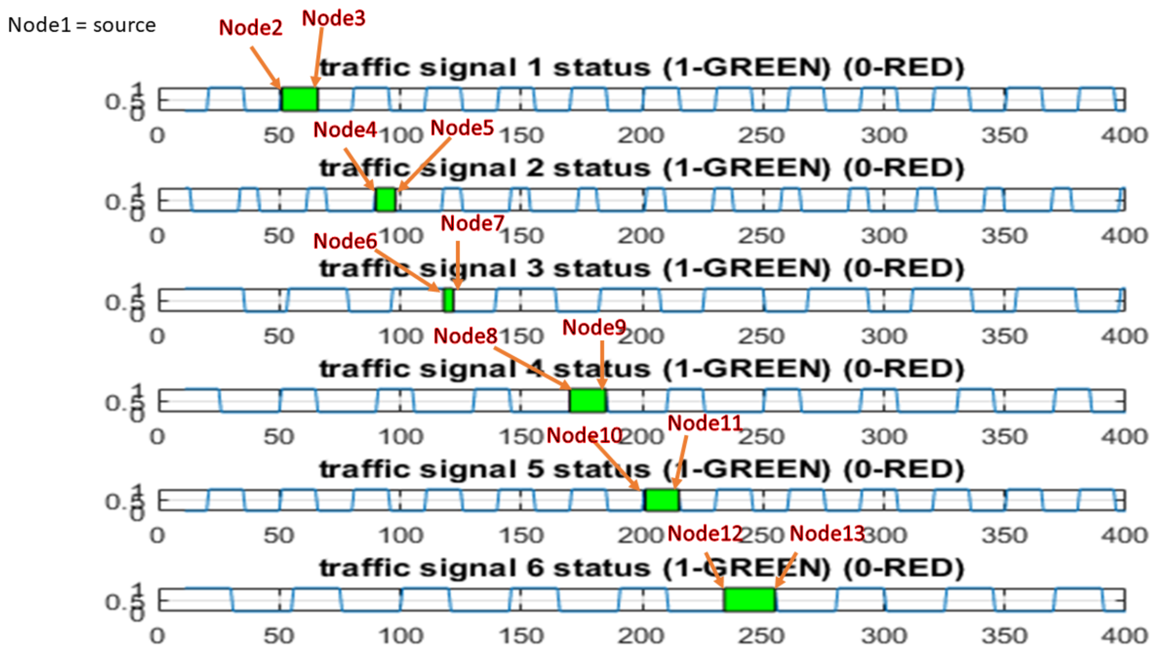


Figure 3.11: Node structure for Dijkstra algorithm.

After feeding the t_{max} and t_{min} values into Adjacency matrix code routine it will generate the adjacency matrix figure 3.12.

13x13 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0.0104	0.0150	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.0094	0.0137	0	0	0	0	0	0	0	0
3	0	0	0	0.0070	0.0098	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0.0067	0.0098	0	0	0	0	0	0
5	0	0	0	0	0	0.0072	0.0070	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0.0104	0.0150	0	0	0	0
7	0	0	0	0	0	0	0	0.0106	0.0152	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0.0064	0.0089	0	0
9	0	0	0	0	0	0	0	0	0	0.0049	0.0065	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0.0097	0.0138
11	0	0	0	0	0	0	0	0	0	0	0	0.0105	0.0098
12	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.12: Adjacency Matrix.

The Adjacency matrix is used to find the lowest weighted path by using Dijkstra algorithm. The results of Dijkstra algorithm are shown in figure 3.13. Where the lowest weighted path in terms of fuel weight is $Node1->2->4->6->8->11->13$.

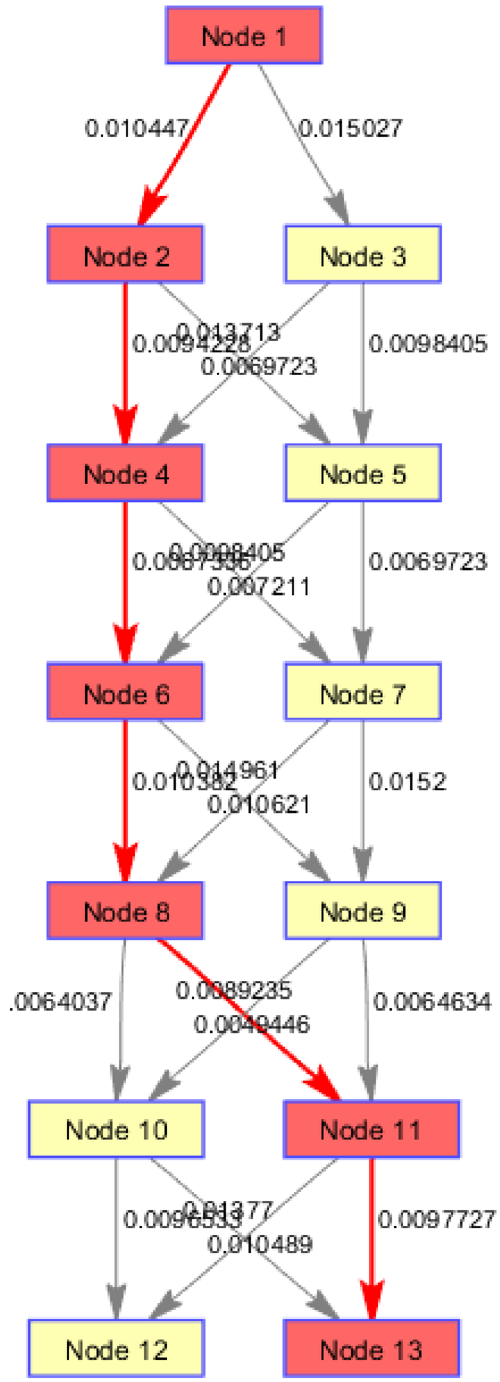


Figure 3.13: Directed graph with fuel weights.

CHAPTER IV

Integration of ”nonstop” model to distance based long term optimization

The optimization model that generates optimal speed profile as described in the previous chapters can be called as ”$Nonstop$” model. This model generates an optimal speed profile; When a vehicle follows the generated speed profile it can drive through traffic signal intersections with out stopping. Thus saving fuel by minimizing excessive acceleration and idling at traffic signal stops.

In the following sub sections we will look into integrated and standalone simulation results of ”$Nonstop$” model.

4.1 Distance based Eco driving scheme long term optimization

The paper ”A Distance-Based Two-Stage Ecological Driving System Using an Estimation of Distribution Algorithm and Model Predictive Control”[2] extends a two-stage ecological driving system optimized in a distance domain by using an EDA and MPC, which consists of a long-term optimization stage and a local adaptation stage. The former stage optimizes the speed profile for an entire trip route without consideration for traffic conditions before departure. The latter stage utilizes the optimal speed profile given by the former stage as a reference speed and adapts the reference speed for a short horizon according to actual traffic conditions while driving. In order to localize the change in the optimal speed profile due to traffic conditions, the optimization was performed in a distance domain.

The long-term optimization[2] cost function is given below:

$$J_{\text{opt}} = w_1 \sum_{k=0}^{n-1} \dot{m}_f(k) \Delta t_k + w_2 \sum_{k=0}^{n-1} \left(v(k+1)^2 - V_{\text{target}}\,(k+1)^2 \right)^2 \tag{4.1}$$

where $\dot{m}_f$ is fuel rate model,w_1,w_2 are weight that can be varied based on the driver's preferences. $v(k+1)$ and $v_{\text{target}}(k+1)$ are the vehicle current speed and target speeds[2]. The long-term optimization doesn't consider the SPAT information to schedule the optimal speed.

The "$Nonstop$" model proposed in this thesis paper can narrow down the speed profile using the SPAT information and this data can be fed to the $v_{\text{target}}(k+1)$ for further optimization. As the proposed model takes less computational time over short horizons it's favorable for online implementation and integration as a feature to the existing optimal solutions.

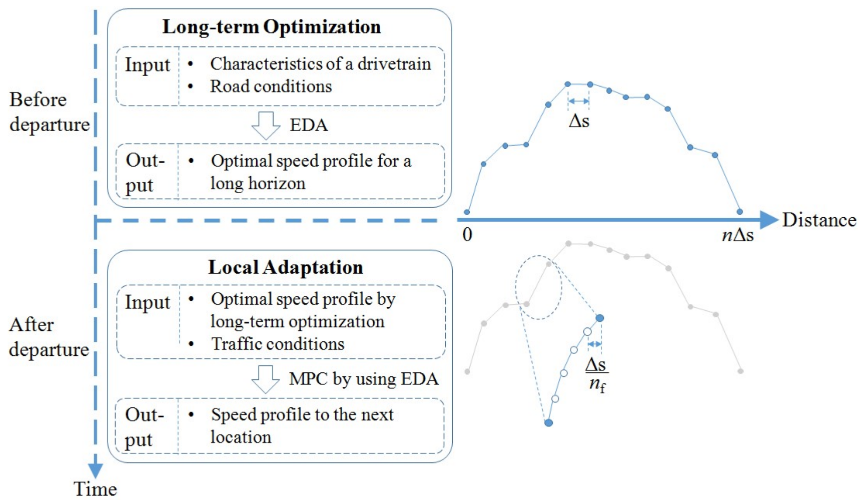


Figure 4.1: Two-Stage Ecological Driving Scheme[2].

Let's look in detail into distance-based two-stage ecological driving system[2]. Figure 4.1 gives an overview of this approach. The eco driving system is divided into two stages first stage is long term optimization. In this stage the speed is optimized for the entire horizon by considering characteristic of drive train and road slopes and speed limits. This optimization is performed before departure. When the vehicle is on move the local optimization comes into picture. In local optimization the speed is followed by considering the optimal speed generated in the long term optimization along with traffic conditions. In areas under heavy traffic conditions the speed is optimized considering the distance from the preceding vehicle.

In a speed profile defined in a time domain, any deviation from the optimal speed profile due to traffic conditions changes the entire reference speed after that time, which requires re-optimization for the whole remaining route. By comparison, in a speed profile defined in a distance domain, any deviation affects only areas under heavy traffic conditions and the reference speeds of other areas are still effective. That is the reason why this optimization is performed in a distance domain[2]. For optimization of cost function authors used EDA algorithm.

4.2 Simulation and Results

For this simulation the vehicle path trajectory is considered to be straight line and has traffic lights at different locations "d_i" mts. The matrix d_i consists of distance of 6 traffic lights from the origin. The "$SPAT$" information in sec, phase cycles k and distance data for each 6 traffic lights are as follows:

$$T_i^0 = [20, 5, 10, 10, 20, 10]\ sec$$
$$T_i = [15, 8, 25, 15, 15, 20]\ sec$$
$$T_i^G = [15, 8, 25, 15, 15, 20]\ sec$$
$$k = 100$$
$$d_i = [1200, 2000, 2900, 3800, 4500, 5000]\ mts$$

The maximum speed limit is $[16, 25, 25, 21, 16, 24]$, V_{min} and vehicle flow V_{flow} are as follows:

$$V_{min} = 1.38\ mts/sec$$
$$V_{flow} = [16, 25, 25, 21, 16, 24]\ mts/sec$$

The vehicle parameters and fuel model are derived from "A Distance-Based Two-Stage Ecological Driving System Using an Estimation of Distribution Algorithm and Model Predictive Control"[2]. The above mentioned parameters are used to determine the optimal speed profile using the $psedocode1$. The standalone and integrated simulations results are discussed in the following sections.

4.2.1 Standalone Simulation Results of proposed model:

After running the $psedocode1$ in matlab environment($intel\ i7\ 4700HQ\ CPU\ core\ 12GB\ RAM$) using the above mentioned parameters the standalone results are as following:

$$Simulation\ time : 383\ ms\ or\ 0.383 sec$$

$$Fuel\ consumed : 0.84\ kg$$

$$Total\ travel\ duration : 642\ sec$$

The simulated graph is relatively large with 205 nodes;

$$Graph\ optimal\ path : NODE\ id's\ 1- > 5- > 60- > 98- > 127- > 158- > 205$$

$$speed\ profile\ (mt/sec)\ between\ above\ node\ Id's = [9.32, 16.14, 38.04, 30.32, 19.42, 1.33]$$

4.2.2 Simulation results when the $v_{\text{target}}\,(k+1)$ is fed with above optimal speed profile:

The speed profile from the standalone model is incorporated as a $v_{\text{target}}\,(k+1)$ for the distanced based long term optimization model suing EAD algorithm[2]. The speed limits for each step 25 mts is distributed in excel using the optimal speed between two intersections. As this model is supposed to work as a speed following model; The $w1$ and $w2$ are swapped. So, that the speed follows the speed profile generated by the standalone model. The simulation results are as follows:

Speed and time profile:

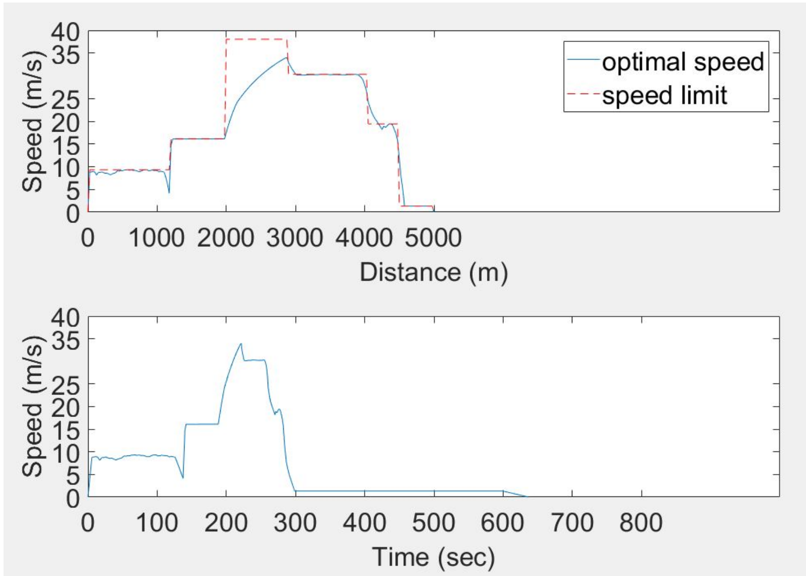


Figure 4.2: speed and time profile.

Torque and brake profile:

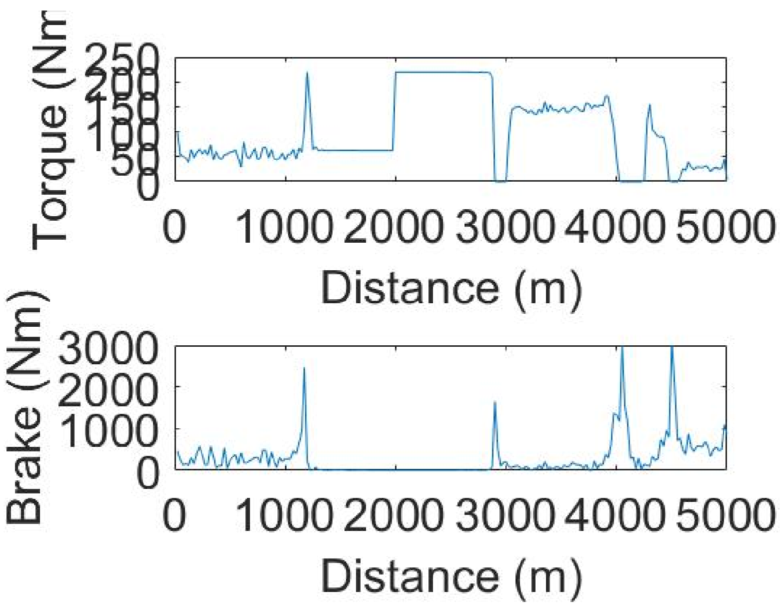


Figure 4.3: torque and brake profile.

$$Simulation\ time : 41.84\ sec$$

$$Fuel\ consumed : 0.36\ kg$$

$$Total\ travel\ duration : 640\ sec$$

4.2.3 Case Studies

In this section we will look into results by varying different parameters.

4.2.4 Case:1

In this case we will simulate the results when the traffic flow V_{flow} is constant throughout trajectory and for all three cases we consider the total distance to be 4500mts. For the last 500mts we assume the vehicle travels at the same speed as of it crossed the 6th traffic light.

Parameters used:

$$V_{min} = 14 \; mts/sec$$
$$V_{speed_limit} = 35 \; mts/sec$$
$$V_{flow} = [25, 25, 25, 25, 25, 25] \; mts/sec$$
$$d_i = [800, 1500, 2200, 3000, 3500, 4500] \; mts$$

The results for cross windows regions are as follows:

Traffic light 1:

$$t_{1,1}^{\min} = 32 \; sec$$
$$t_{1,2}^{\max} = 36 \; sec$$

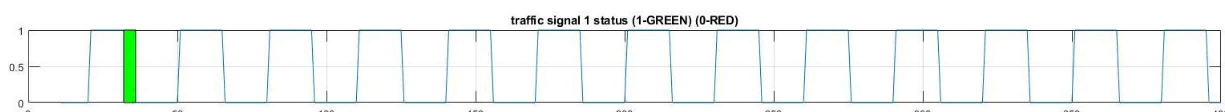


Figure 4.4: Case1 Crossing window for Traffic light 1. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 2:

$$t_{2,1}^{\min} = 62 \; sec$$
$$t_{2,2}^{\max} = 70 \; sec$$

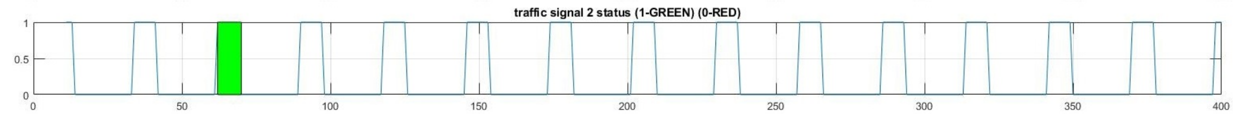


Figure 4.5: Case1 Crossing window for Traffic light 2. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 3:

$$t_{3,1}^{\min} = 97 \; sec$$

$$t_{3,2}^{\max} = 122 \; sec$$

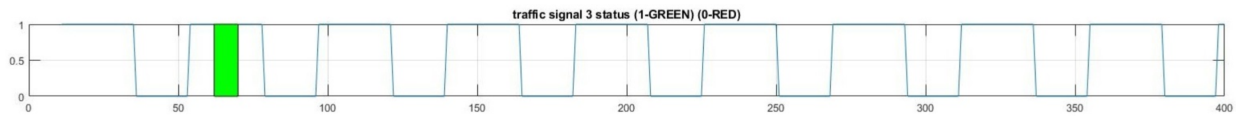


Figure 4.6: Case1 Crossing window for Traffic light 3. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 4:

$$t_{4,1}^{\min} = 131 \; sec$$

$$t_{4,2}^{\max} = 146 \; sec$$

$$t_{4,3}^{\min} = 171 \; sec$$

$$t_{4,4}^{\max} = 179.14 \; sec$$

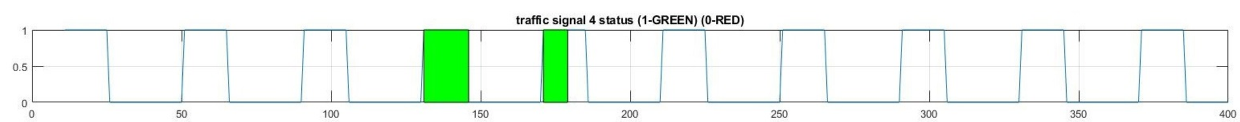


Figure 4.7: Case1 Crossing window for Traffic light 4. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 5:

$$t_{5,1}^{\min} = 151 \; sec$$

$$t_{5,2}^{\max} = 156 \; sec$$

$$t_{5,3}^{\min} = 181 \; sec$$

$$t_{5,4}^{\max} = 181.71 \; sec$$

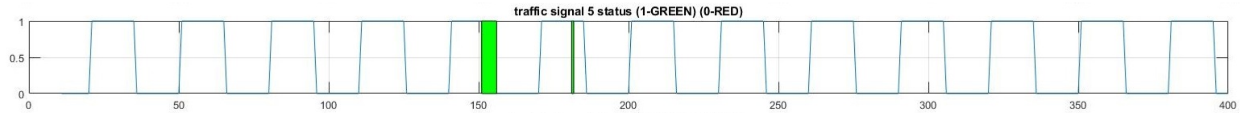


Figure 4.8: Case1 Crossing window for Traffic light 5. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 6:

$$t_{6,1}^{\min} = 191 \; sec$$

$$t_{6,2}^{\max} = 211 \; sec$$

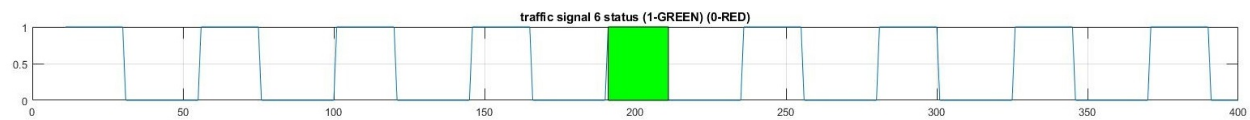


Figure 4.9: Case1 Crossing window for Traffic light 6. x-axis: time(sec), y-axis:signal status(1 or 0).

In order for the vehicle to reach the destination without stopping; The vehicle has to cross the intersection during the green windows figure 4.10 . By using the optimal path approach the vehicle not only crosses the green windows but also it can be optimized to find the best fuel economical path.

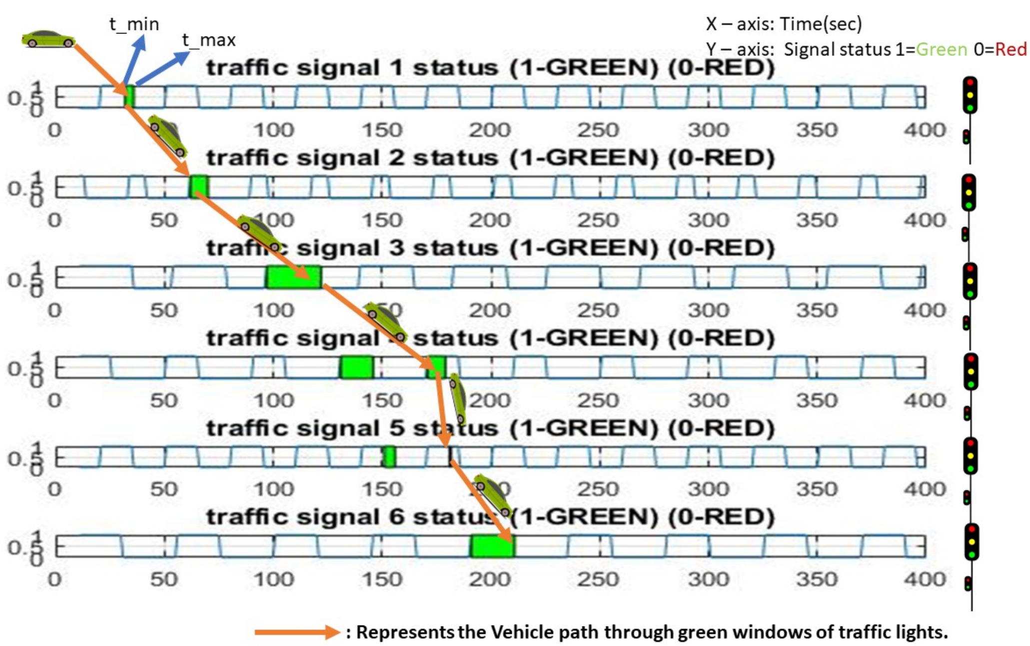


Figure 4.10: Case1 Vehicle trajectory through green crossing windows.

For the optimization using Dijkstra algorithm each t_{max} and t_{min} values for 6 traffic lights are considered as individual nodes as shown in figure 4.11.

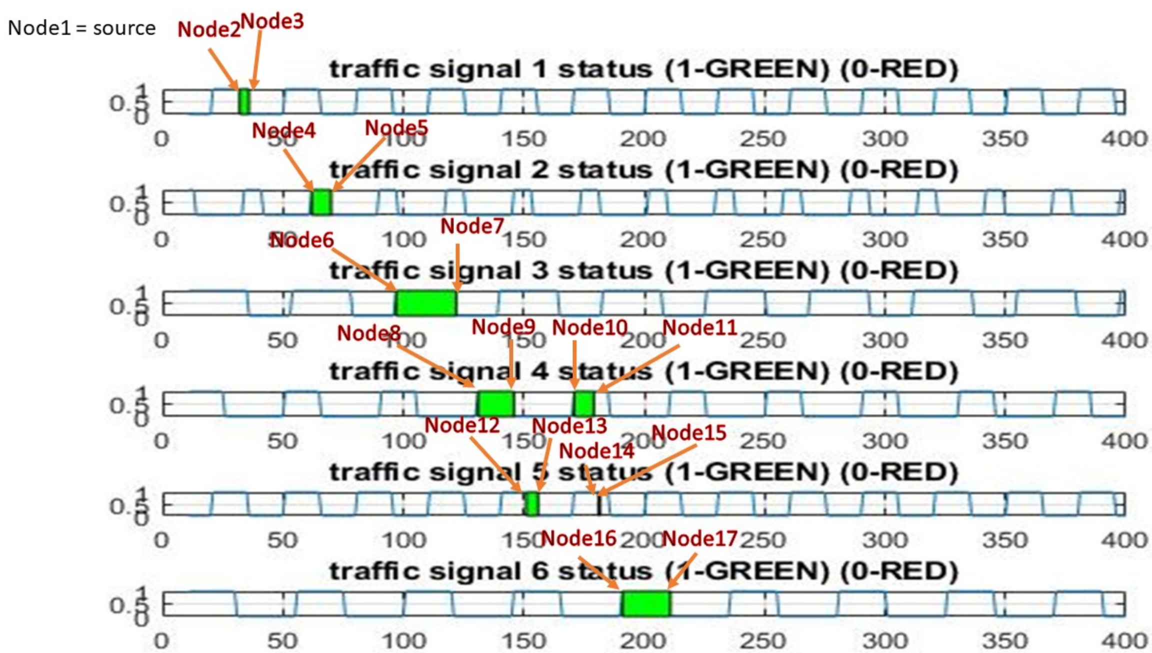


Figure 4.11: Case1 Node structure for Dijkstra algorithm.

After feeding the t_{max} and t_{min} values into Adjacency matrix code routine it will generate the adjacency matrix figure 4.13

17x17 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0.0077	0.0113	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0.0100	0.0095	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0.0069	0.0097	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0.0097	0.0129	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0.0068	0.0134	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0.0115	0.0106	0.0145	0.0141	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0.0091	0.0082	0.0106	0.0156	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	0.0047	0.0068	0.0086	0.0086	0	0	
9	0	0	0	0	0	0	0	0	0	0	0	0.0056	0.0053	0.0095	0.0095	0	0	
10	0	0	0	0	0	0	0	0	0	0	0	0.0303	0.0300	0.0053	0.0053	0	0	
11	0	0	0	0	0	0	0	0	0	0	0	0.0308	0.0305	0.0058	0.0058	0	0	
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0098	0.0134	
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0101	0.0137	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0115	0.0103	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0116	0.0104	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18																		

Figure 4.12: Case1 Adjacency Matrix.

The Adjacency matrix is used to find the lowest weighted path by using Dijkstra algorithm. The results of Dijkstra algorithm are shown in figure 4.13 . Where the lowest weighted path in terms of fuel weight is $Node1 - 2 - 5 - 6 - 9 - 13 - 17$.

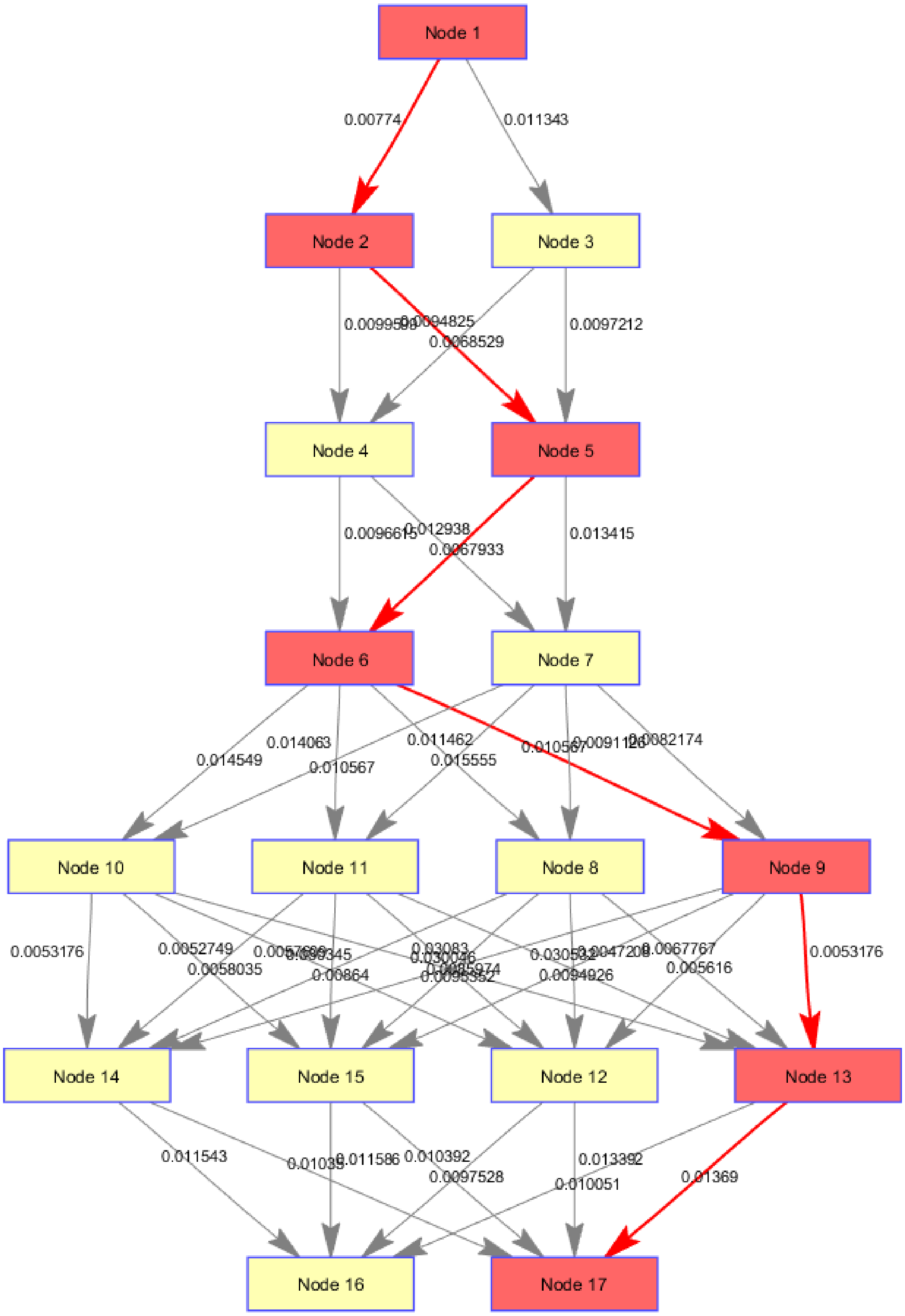


Figure 4.13: Case1 Directed graph with fuel weights.

After running the $psedocode1$ in matlab environment($intel\ i7\ 4700HQ\ CPU\ core\ 12GB\ RAM$) using the above mentioned parameters the standalone results are as following:

$$Simulation\ time : 71\ ms\ or\ 0.071 sec$$

$$Fuel\ consumed : 0.38\ kg$$

$$Total\ travel\ duration : 217.95\ sec$$

The simulated graph has 17 nodes;

$$Graph\ optimal\ path : NODE\ id's\ 1->2->5->6->9->13->17$$

$$speed\ profile\ (mt/sec)\ between\ above\ node\ Id's = [24.21, 17.76, 25, 15.81, 47.5, 17.72]$$

The speed profile from the standalone model is incorporated as a $v_{\text{target}}(k+1)$ for the distanced based long term optimization model suing EAD algorithm[2]. The speed limits for each step 25 mts is distributed in excel using the optimal speed between two intersections. As this model is supposed to work as a speed following model; The $w1$ and $w2$ are swapped. So, that the speed follows the speed profile generated by the standalone model. The simulation results are as follows:

Speed and time profile:

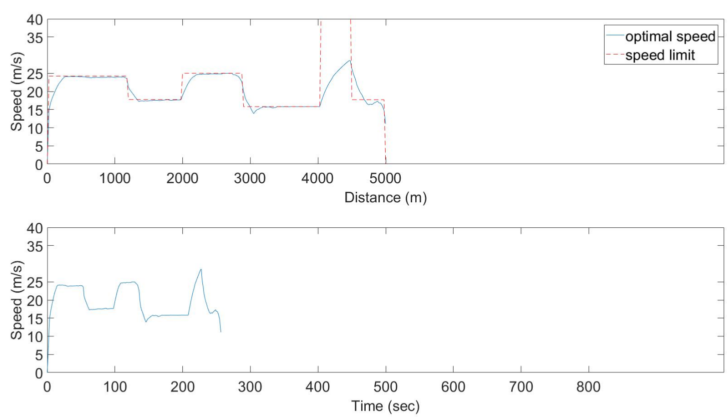


Figure 4.14: Case1 speed and time profile.

Torque and brake profile:

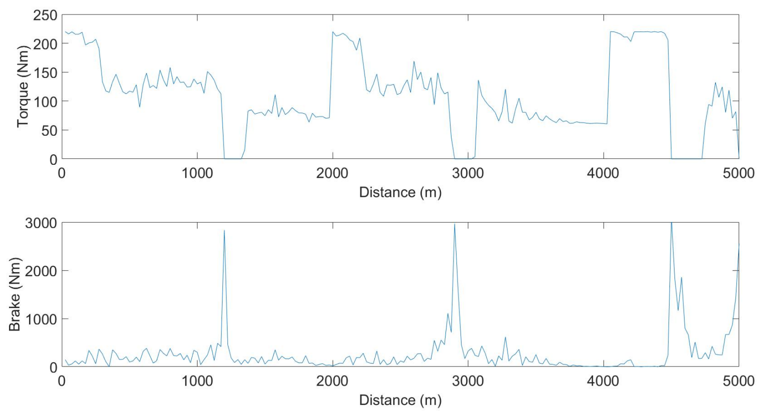


Figure 4.15: Case1 torque and brake profile.

$$Simulation\ time : 30.73\ sec$$

$$Fuel\ consumed : 0.35\ kg$$

$$Total\ travel\ duration : 256.53\ seconds$$

In the Case1 we observed that with the V_{flow} of traffic being constant the fuel consumed is less, travel time is less, simulation time is improved.

case2

4.2.5 Case:2

In this case we will simulate the results when the traffic flow V_{flow} is closer to V_{max_limit} throughout trajectory.

Parameters used:

$$V_{min} = 14\ mts/sec$$

$$V_{speed_limit} = 35\ mts/sec$$

$$V_{flow} = [30, 30, 30, 30, 30, 30]\ mts/sec$$

$$V_{max_limit} = [35, 35, 35, 35, 35, 35]\ mts/sec$$

$$d_i = [800, 1500, 2200, 3000, 3500, 4500]\ mts$$

The results for cross windows regions are as follows:

Traffic light 1:

$$t_{1,1}^{\min} = 26.66\ sec$$

$$t_{1,2}^{\max} = 35.66\ sec$$

$$t_{1,3}^{\min} = 56.66\ sec$$

$$t_{1,4}^{\max} = 57.14\ sec$$

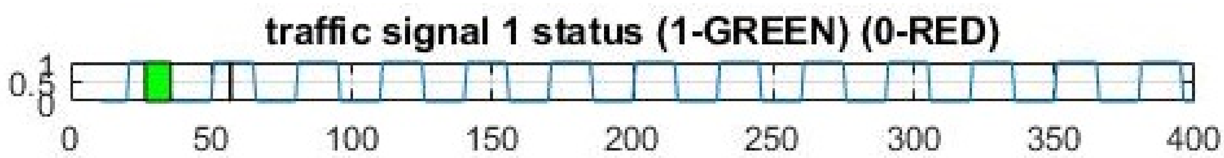


Figure 4.16: Case2 Crossing window for Traffic light 1. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 2:

$$t_{2,1}^{\min} = 62\ sec$$

$$t_{2,2}^{\max} = 70\ sec$$

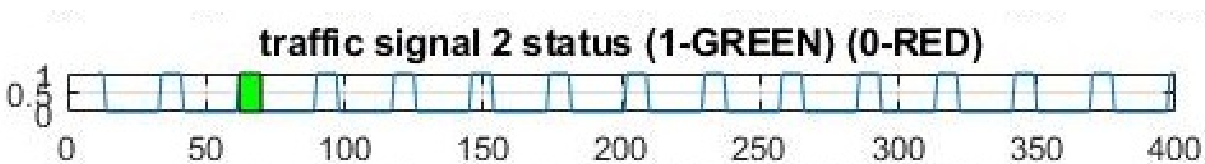


Figure 4.17: Case2 Crossing window for Traffic light 2. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 3:

$$t_{3,1}^{\min} = 96.33\ sec$$

$$t_{3,2}^{\max} = 121.33\ sec$$

Traffic light 4:

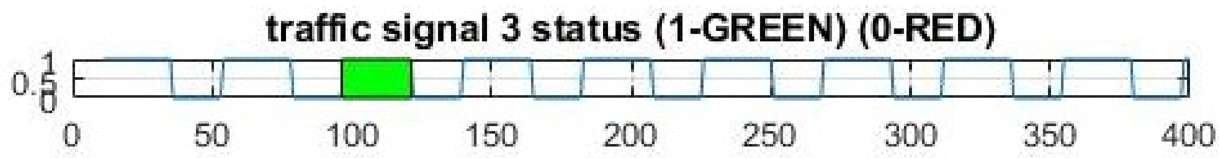


Figure 4.18: Case2 Crossing window for Traffic light 3. x-axis: time(sec), y-axis:signal status(1 or 0).

$$t_{4,1}^{\min} = 131 \; sec$$

$$t_{4,2}^{\max} = 146 \; sec$$

$$t_{4,3}^{\min} = 171 \; sec$$

$$t_{4,4}^{\max} = 178.47 \; sec$$

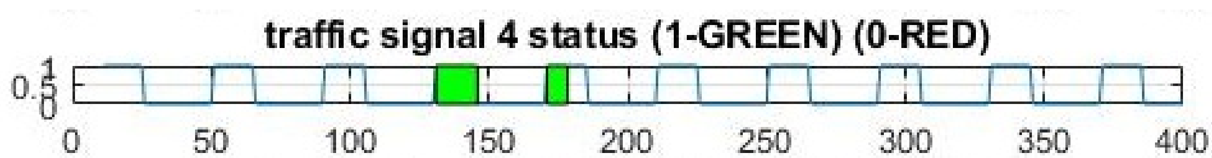


Figure 4.19: Case2 Crossing window for Traffic light 4. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 5:

$$t_{5,1}^{\min} = 147.66 \; sec$$

$$t_{5,2}^{\max} = 155.66 \; sec$$

$$t_{5,3}^{\min} = 177.66 \; sec$$

$$t_{5,4}^{\max} = 181.71 \; sec$$

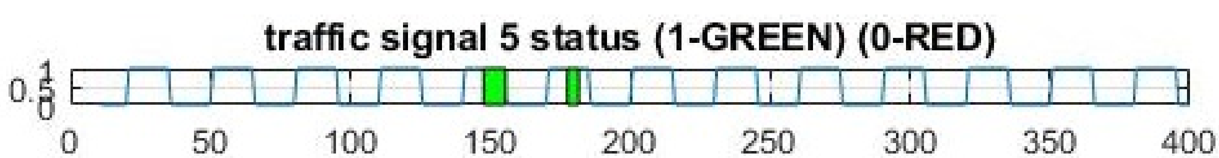


Figure 4.20: Case2 Crossing window for Traffic light 5. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 6:

$$t_{6,1}^{\min} = 191 \; sec$$

$$t_{6,2}^{\max} = 211 \; sec$$

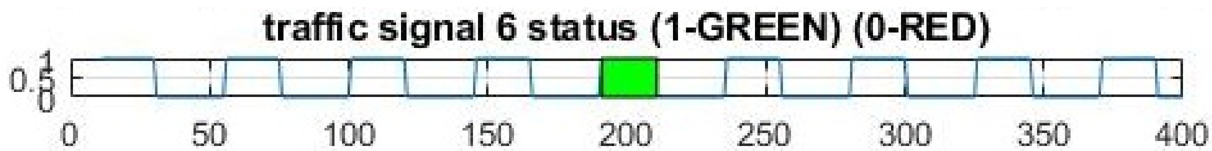


Figure 4.21: Case2 Crossing window for Traffic light 6. x-axis: time(sec), y-axis:signal status(1 or 0).

In order for the vehicle to reach the destination without stopping; The vehicle has to cross the intersection during the green windows figure 4.22 . By using the optimal path approach the vehicle not only crosses the green windows but also it can be optimized to find the best fuel economical path.

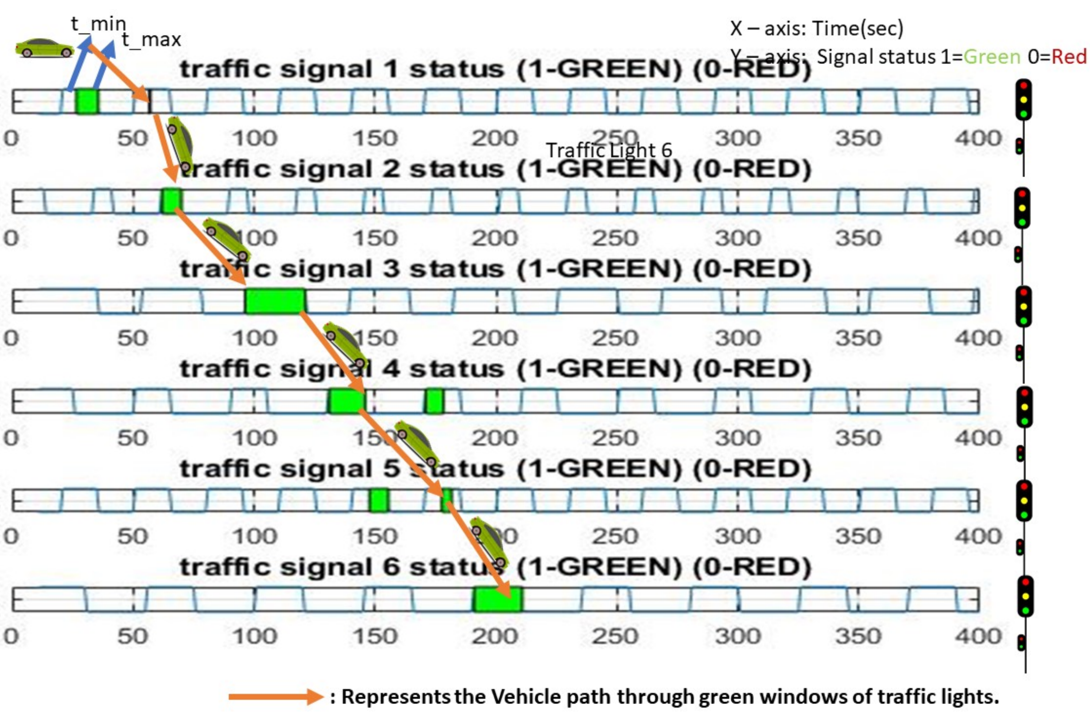


Figure 4.22: Case2 Vehicle trajectory through green crossing windows.

For the optimization using Dijkstra algorithm each t_{max} and t_{min} values for 6 traffic lights are considered as individual nodes as shown in figure 3.11.

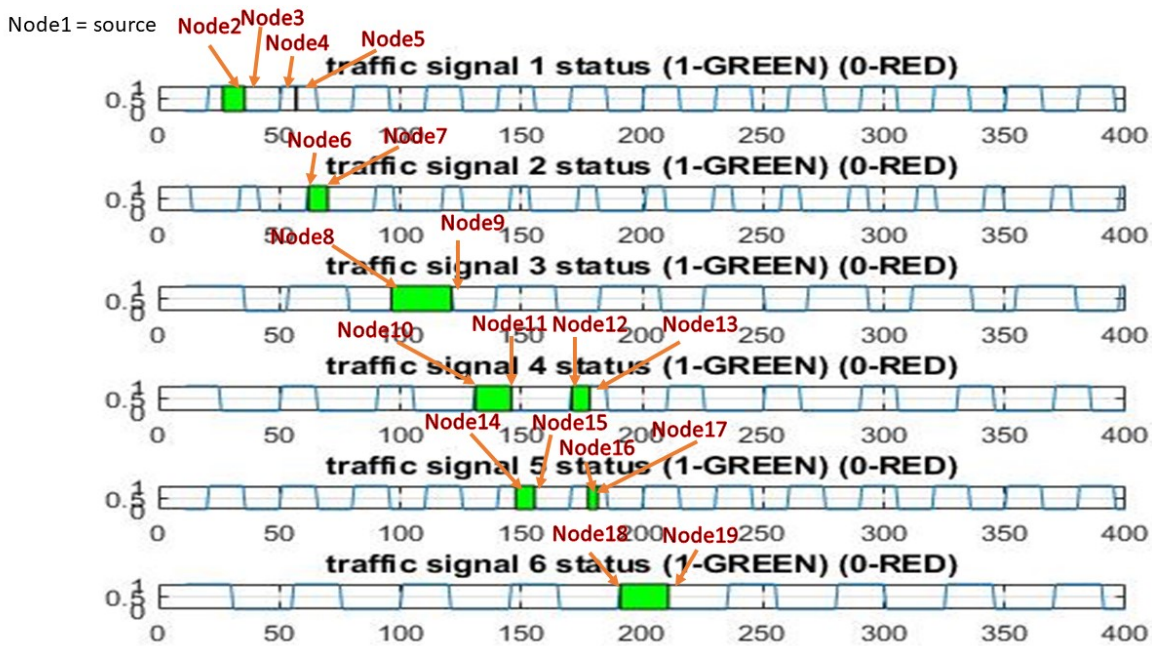
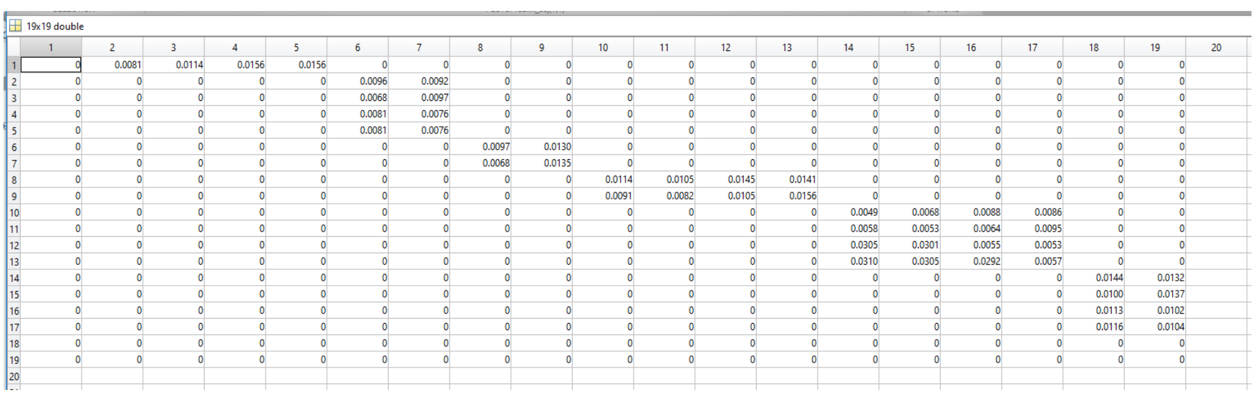


Figure 4.23: Case2 Node structure for Dijkstra algorithm.

After feeding the t_{max} and t_{min} values into Adjacency matrix code routine it will generate the adjacency matrix figure 4.13


19x19 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0.0081	0.0114	0.0156	0.0156	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0.0096	0.0092	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0.0068	0.0097	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0.0081	0.0076	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0.0081	0.0076	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0.0097	0.0130	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0.0068	0.0135	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0.0114	0.0105	0.0145	0.0141	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0.0091	0.0082	0.0105	0.0156	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0049	0.0068	0.0088	0.0086	0	0	
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0058	0.0053	0.0064	0.0095	0	0	
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0305	0.0301	0.0055	0.0053	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0310	0.0305	0.0292	0.0057	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0144	0.0132	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0100	0.0137	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0113	0.0102	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0116	0.0104	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20																				

Figure 4.24: Case2 Adjacency Matrix.

The Adjacency matrix is used to find the lowest weighted path by using Dijkstra algorithm. The results of Dijkstra algorithm are shown in figure 4.25 . Where the lowest weighted path in terms of fuel weight is $Node1 - 2 - 7 - 8 - 11 - 16 - 19$.

$$Simulation\ time : 59\ ms\ or\ 0.0599 sec$$

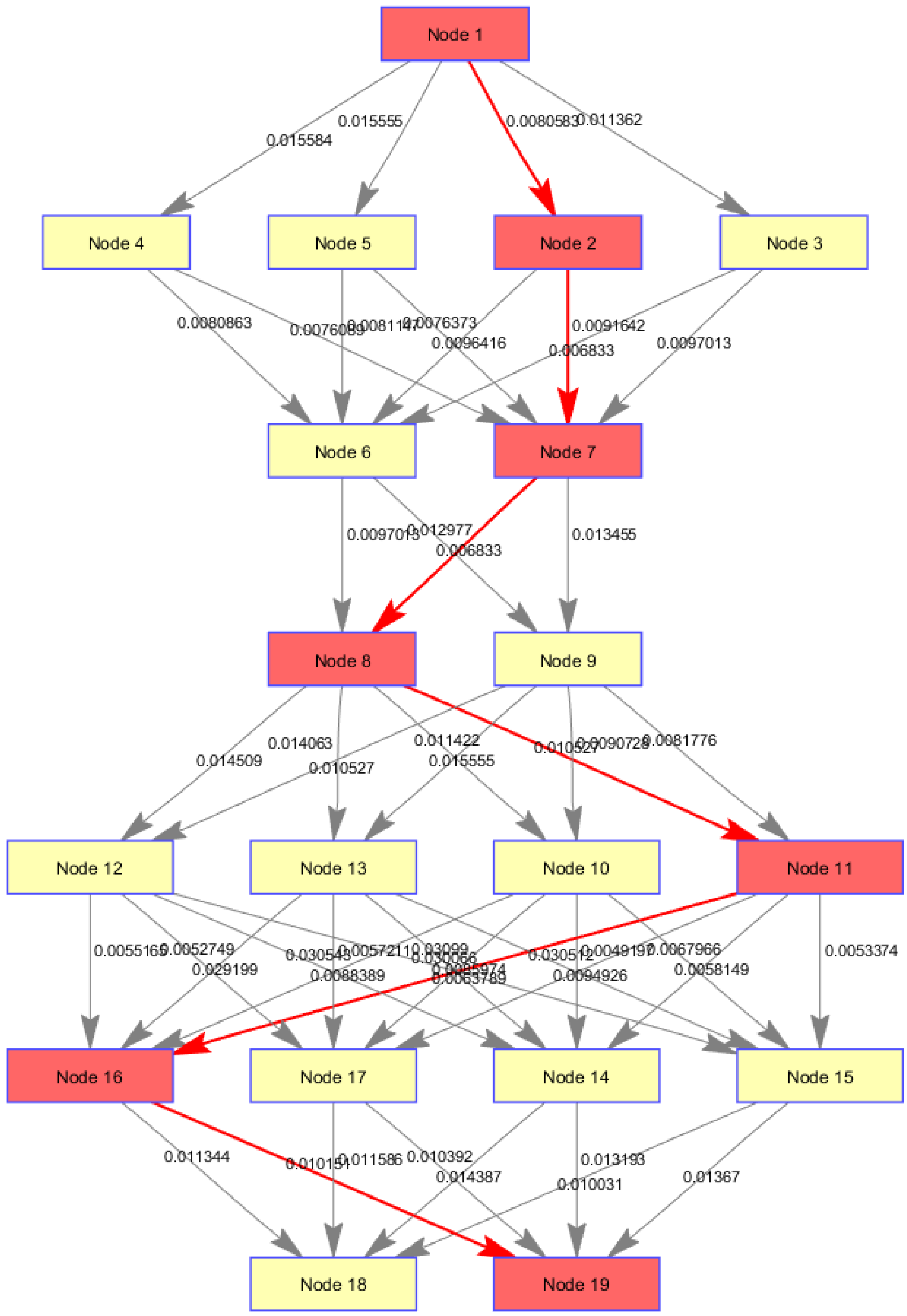


Figure 4.25: Case2 Directed graph with fuel weights.

$$Fuel\ consumed : 0.41\ kg$$

$$Total\ travel\ duration : 218.56\ sec$$

The simulated graph is of 19 nodes;

$$Graph\ optimal\ path : NODE\ id's\ 1- > 2- > 7- > 8- > 11- > 16- > 19$$

$$speed\ profile\ (mt/sec)\ between\ above\ node\ Id's = [29.06, 15.57, 25.63, 15.60, 15, 29.25]$$

The speed profile from the standalone model is incorporated as a $v_{\text{target}}\,(k+1)$ for the distanced based long term optimization model suing EAD algorithm[2]. The speed limits for each step 25 mts is distributed in excel using the optimal speed between two intersections. As this model is supposed to work as a speed following model; The $w1$ and $w2$ are swapped. So, that the speed follows the speed profile generated by the standalone model. The simulation results figures are as follows:

Speed and time profile:

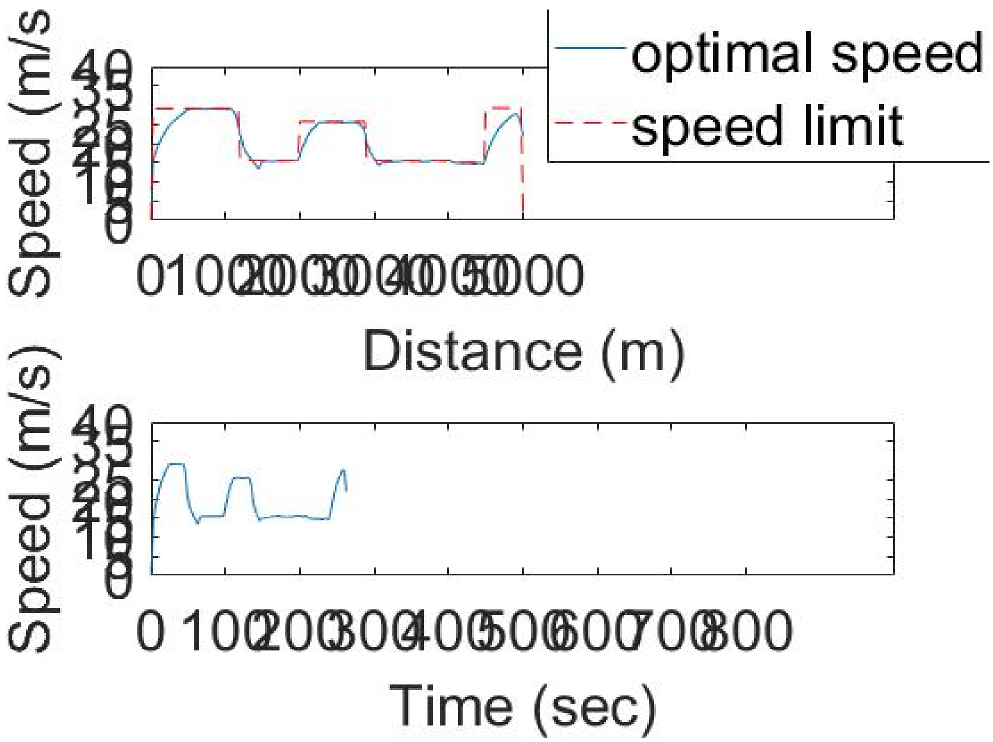


Figure 4.26: Case2 speed and time profile.

Torque and brake profile:

$$Simulation\ time : 23.31\ sec$$

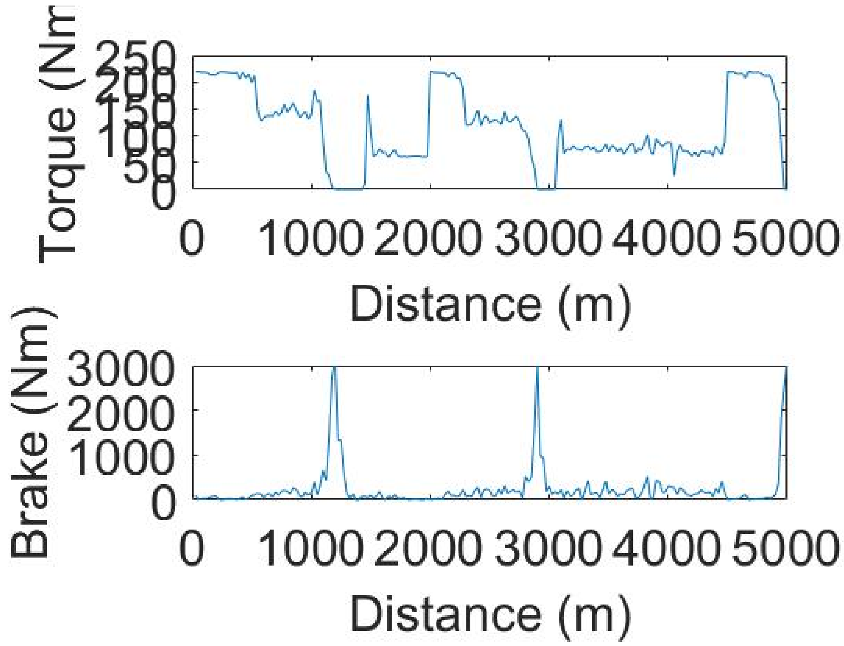


Figure 4.27: Case2 torque and brake profile.

$$Fuel\ consumed : 0.36\ kg$$

$$Total\ travel\ duration : 262.20\ seconds$$

In the Case2 we observed that with the V_{flow} of traffic being constant the fuel consumed is less, travel time is less, simulation time is improved.

4.2.6 Case:3

In this case we will simulate the results when the traffic flow T_i^G is constant throughout trajectory while considering the case2 conditions for traffic conditions.

Parameters used:

$$T_i^G = [10, 10, 10, 10, 10, 10]\ sec$$

$$V_{min} = 14\ mts/sec$$

$$V_{speed_limit} = 35\ mts/sec$$

$$V_{flow} = [30, 30, 30, 30, 30, 30]\ mts/sec$$

$$d_i = [800, 1500, 2200, 3000, 3500, 4500]\ mts$$

The results for cross windows regions are as follows:

Traffic light 1:

$$t_{1,1}^{\min} = 26.66\ sec$$

$$t_{1,2}^{\max} = 35.66\ sec$$

$$t_{1,3}^{\min} = 51.66\ sec$$

$$t_{1,4}^{\max} = 57.14\ sec$$

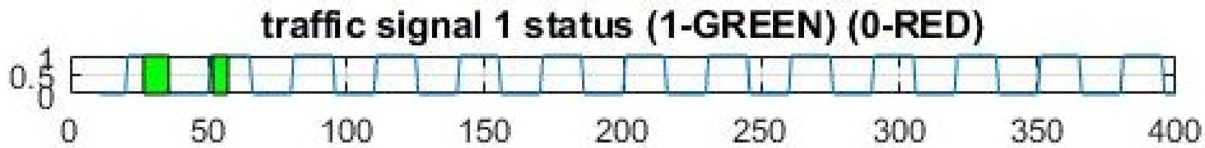


Figure 4.28: Case3 Crossing window for Traffic light 1. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 2:

$$t_{2,1}^{\min} = 60\ sec$$

$$t_{2,2}^{\max} = 70\ sec$$

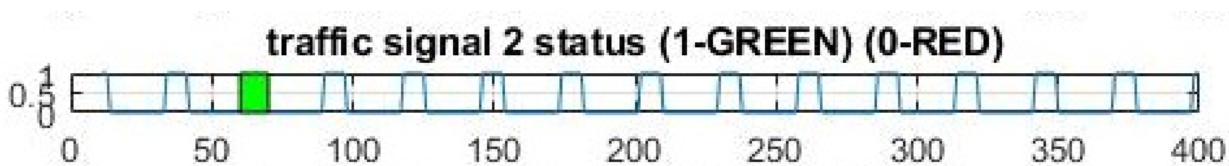


Figure 4.29: Case3 Crossing window for Traffic light 2. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 3:

$$t_{3,1}^{\min} = 111.33\ sec$$

$$t_{3,2}^{\max} = 121.33\ sec$$

Traffic light 4:

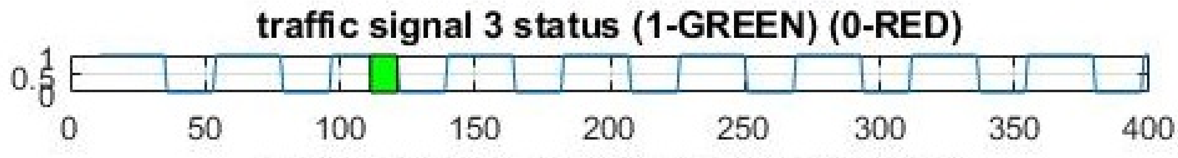


Figure 4.30: Case3 Crossing window for Traffic light 3. x-axis: time(sec), y-axis:signal status(1 or 0).

$$t_{4,1}^{\min} = 138\ sec$$
$$t_{4,2}^{\max} = 146\ sec$$
$$t_{4,3}^{\min} = 173\ sec$$
$$t_{4,4}^{\max} = 178.47\ sec$$

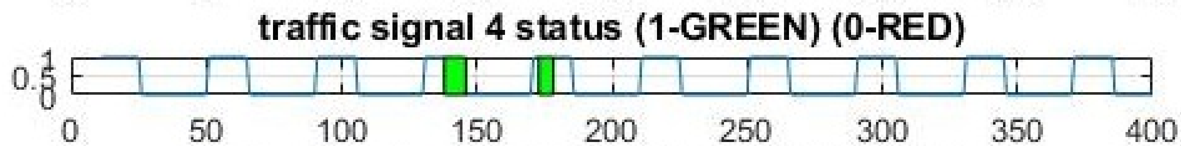


Figure 4.31: Case3 Crossing window for Traffic light 4. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 5:

$$t_{5,1}^{\min} = 154.66\ sec$$
$$t_{5,2}^{\max} = 155.66\ sec$$
$$t_{5,3}^{\min} = 179.66\ sec$$
$$t_{5,4}^{\max} = 181.71\ sec$$

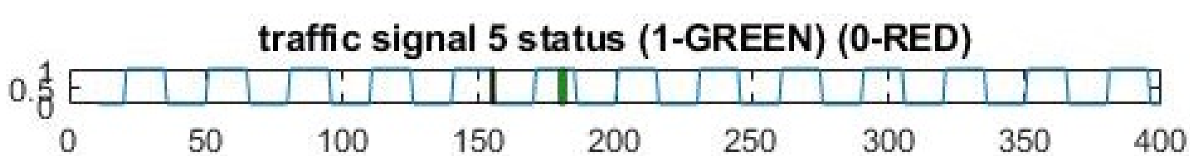


Figure 4.32: Case3 Crossing window for Traffic light 5. x-axis: time(sec), y-axis:signal status(1 or 0).

Traffic light 6:

$$t_{6,1}^{\min} = 201\ sec$$

$$t_{6,2}^{\max} = 211\ sec$$

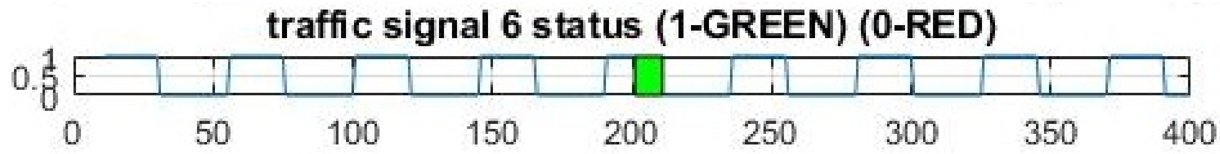


Figure 4.33: Case3 Crossing window for Traffic light 6. x-axis: time(sec), y-axis:signal status(1 or 0).

In order for the vehicle to reach the destination without stopping; The vehicle has to cross the intersection during the green windows figure 4.34 . By using the optimal path approach the vehicle not only crosses the green windows but also it can be optimized to find the best fuel economical path.

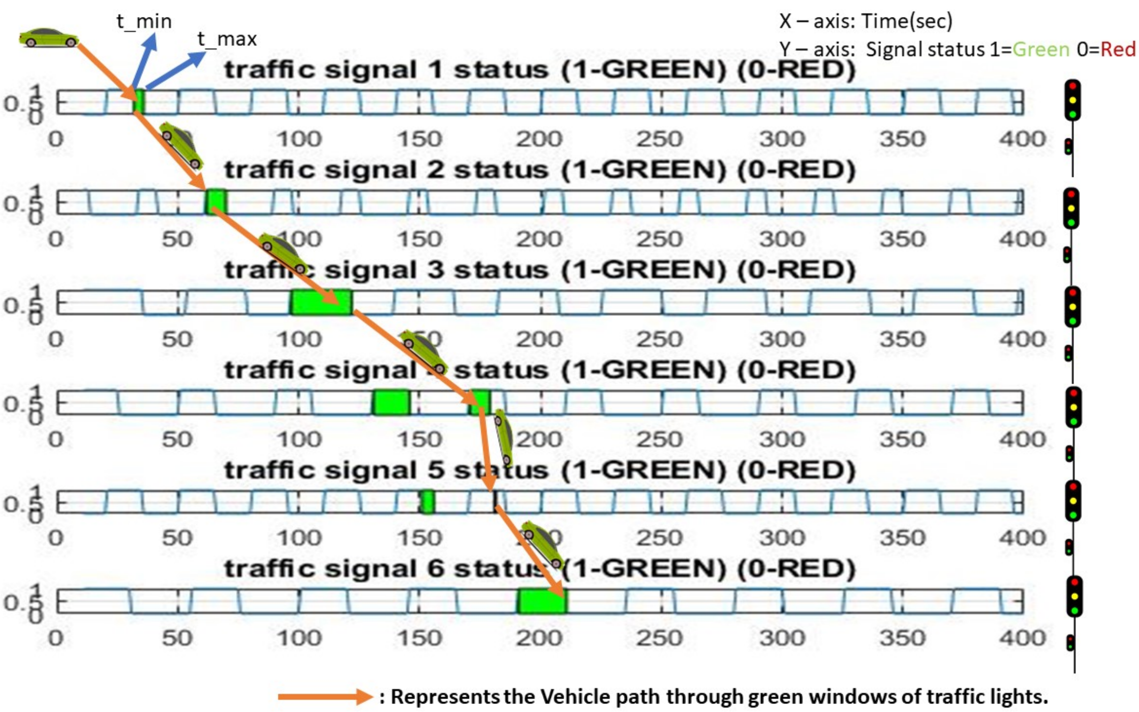


Figure 4.34: Case3 Vehicle trajectory through green crossing windows.

For the optimization using Dijkstra algorithm each t_{max} and t_{min} values for 6 traffic lights are considered as individual nodes as shown in figure 4.35.

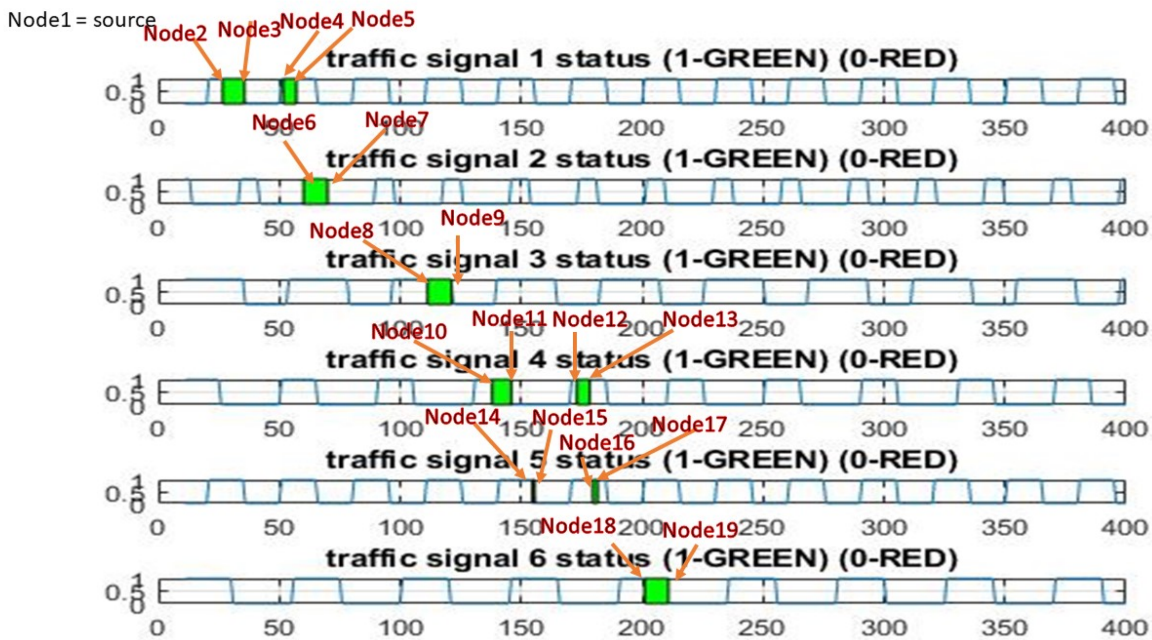


Figure 4.35: Case3 Node structure for Dijkstra algorithm.

After feeding the t_{max} and t_{min} values into Adjacency matrix code routine it will generate the adjacency matrix figure 4.36

19x19 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0.0081	0.0114	0.0104	0.0156	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0.0098	0.0092	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0.0070	0.0097	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0.0079	0.0073	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0.0082	0.0076	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0.0135	0.0129	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0.0093	0.0135	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0.0081	0.0114	0.0153	0.0150	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0.0087	0.0082	0.0104	0.0156	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0049	0.0049	0.0091	0.0090	0	0	
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0054	0.0053	0.0096	0.0095	0	0	
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0302	0.0302	0.0055	0.0054	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0306	0.0305	0.0058	0.0057	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0142	0.0136	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0143	0.0137	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0109	0.0103	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0110	0.0104	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20																				

Figure 4.36: Case3 Adjacency Matrix.

The Adjacency matrix is used to find the lowest weighted path by using Dijkstra algorithm. The results of Dijkstra algorithm are shown in figure 4.37 . Where the lowest weighted path in terms of fuel weight is $Node1 - 2 - 7 - 8 - 10 - 15 - 19$.

After running the $psedocode1$ in matlab environment($intel\ i7\ 4700HQ\ CPU\ core\ 12GB\ RAM$) using the above mentioned parameters the standalone results are as following:

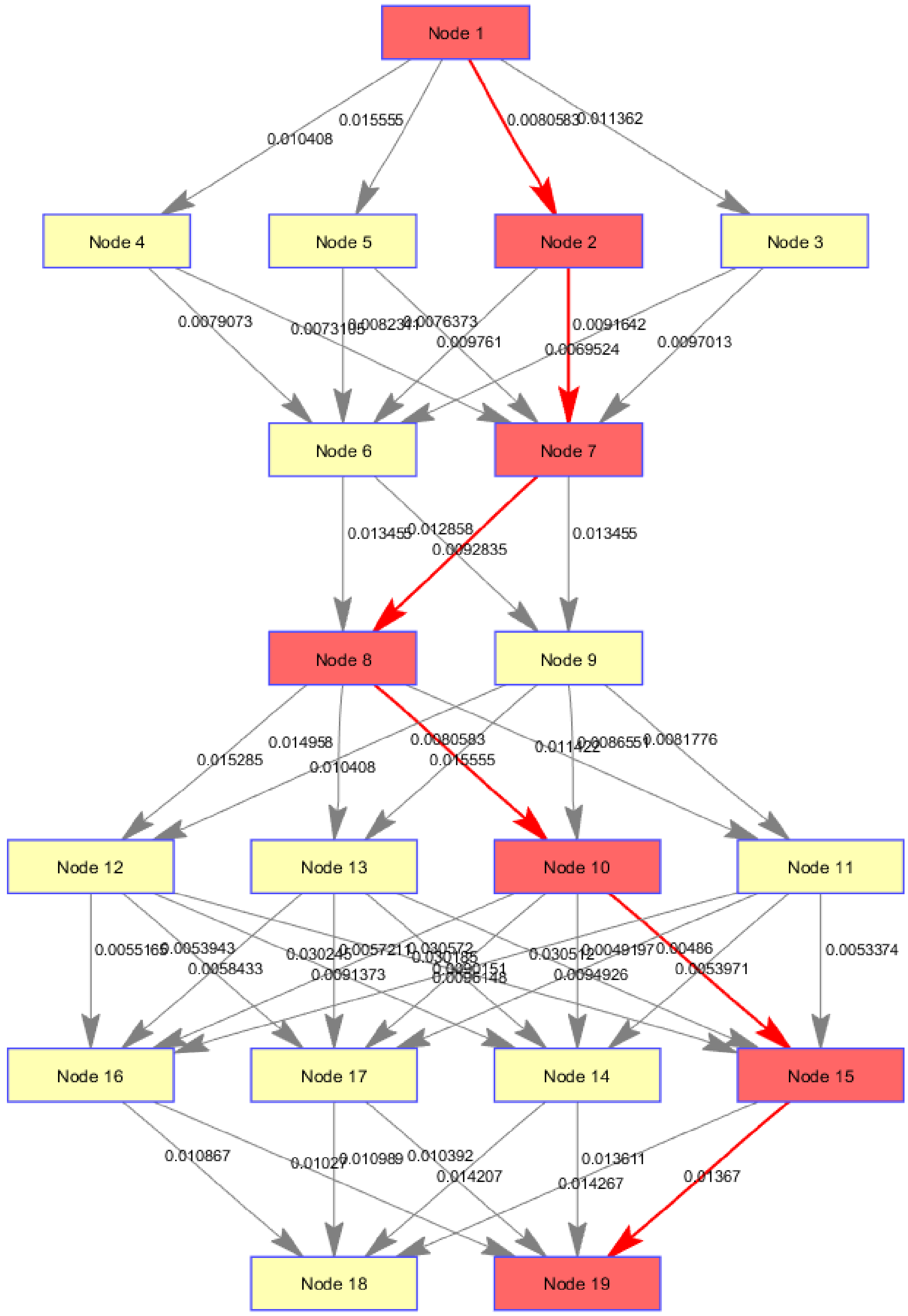


Figure 4.37: Case3 Directed graph with fuel weights.

$$Simulation\ time : 1.12 sec$$

$$Fuel\ consumed : 0.39\ kg$$

$$Total\ travel\ duration : 217.95\ sec$$

The simulated graph has 19 nodes;

$$Graph\ optimal\ path : NODE\ id's\ 1- > 2- > 7- > 8- > 10- > 15- > 19$$

$$speed\ profile\ (mt/sec)\ between\ above\ node\ Id's = [29.06, 15.57, 16.33, 29.06, 26.88, 17.62]$$

The speed profile from the standalone model is incorporated as a $v_{\text{target}}\,(k+1)$ for the distanced based long term optimization model suing EAD algorithm[2]. The speed limits for each step 25 mts is distributed in excel using the optimal speed between two intersections. As this model is supposed to work as a speed following model; The $w1$ and $w2$ are swapped. So, that the speed follows the speed profile generated by the standalone model. The simulation results are as follows:

Speed and time profile:

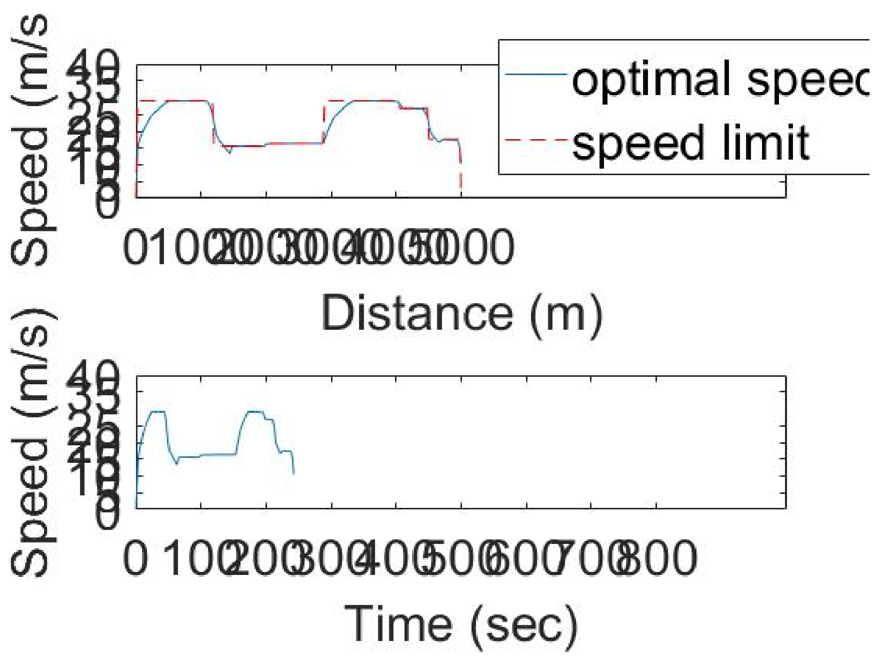


Figure 4.38: Case3 speed and time profile.

Torque and brake profile:

$$Simulation\ time : 25.37\ sec$$

$$Fuel\ consumed : 0.35\ kg$$

$$Total\ travel\ duration : 243.02\ seconds$$

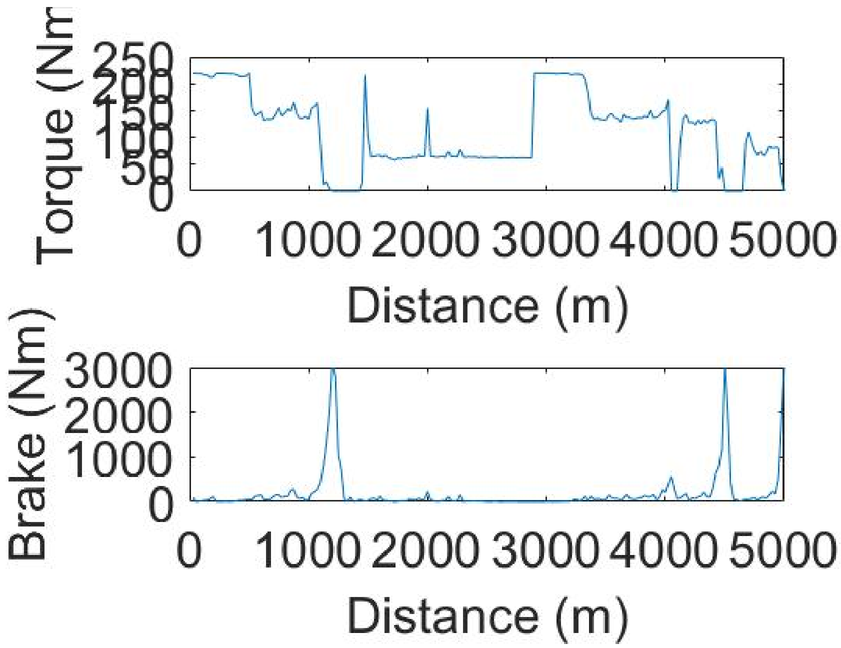


Figure 4.39: Case3 torque and brake profile.

In the Case3 we observed that with the T_i^G of traffic being constant the fuel consumed is less, travel time is less, simulation time is improved compared with case2. As the T_i^G is constant it improves the predictability of traffic environment.

BIBLIOGRAPHY

[1] European Commission Climate Action, “What is the EU doing about climate change?”, https://www.eea.europa.eu/data-and-maps/indicators/greenhouse-gas-emission-trends-6/assessment-1.

[2] H. Lim, C. C. Mi and W. Su, ”A Distance-Based Two-Stage Ecological Driving System Using an Estimation of Distribution Algorithm and Model Predictive Control,” in IEEE Transactions on Vehicular Technology, vol. 66, no. 8, pp. 6663-6675, Aug. 2017.

[3] S. Mandava, K. Boriboonsomsin, M. Barth, “Arterial velocity planning based on traffic signal information under light traffic conditions”, Proc. 12th International IEEE Conference on Intelligent Transportation Systems ITSC ’09, 2009.

[4] Hunt PB, Robertson DI, Bretherton RD, Winton RI. SCOOT - A Traffic Responsive Method of Co-Ordinating Signals. Technical Report, TRRL Laboratory Report 1014 1981.

[5] Diakaki C. Integrated Control of Traffic Flow in Corridor Networks. PhD Thesis, Technical University of Crete 1999.

[6] Urban traffic control systems; www.its.leeds.ac.uk.

[7] V. A. Butakov and P. Ioannou, ”Personalized Driver Assistance for Signalized Intersections Using V2I Communication,” in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 7, pp. 1910-1919, July 2016.

[8] H. Lim, W. Su and C. C. Mi, ”Distance-Based Ecological Driving Scheme Using a Two-Stage Hierarchy for Long-Term Optimization and Short-Term Adaptation,” in IEEE Transactions on Vehicular Technology, vol. 66, no. 3, pp. 1940-1949, March 2017.

[9]https://pxhere.com/en/photo/131869.