



SAPIENZA
UNIVERSITÀ DI ROMA

2020

Designing Signal Setting of Via Prenestina



Using Different Methods for Optimization:

- 1- Brute Force Algorithm**
- 2- GA (Genetic Algorithm)**
- 3- Hill Climbing Algorithm**

Alireza Alipourfaz
Sapienza University
7/27/2020



SAPIENZA
UNIVERSITÀ DI ROMA

TRAFFIC ENGINEERING AND ITS

Report of the Final Traffic Project

Prof. Gaetano Fusco



Presented by:

Alireza Alipourfaz

Contents

1. Introduction.....	5
2. Study Area.....	5
3. Data Collection.....	6
4. Methodology	7
4.1. Input Parameters	9
4.2. Lane Grouping and Demand Flow Rate.....	10
4.2. Saturation Flow Rate	11
4.4. Capacity.....	12
4.5. v/c Ratio	12
4.6. Delay	13
4.7. Level of Service (LOS).....	15
5. Junctions' analysis: Current situation.....	16
5.1.1 Junction 1: Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina	16
5.2.2 Junction 2: Via Dignano d'Istria – Largo Irpinia - via Prenestina - via Prenestina	25
5.3.3 Junction 3: Via Olevano Romano - via Prenestina - via Prenestina	35
6. Optimization Methods	45
6.1. Minimum Cycle	45
6.2. Optimum Cycle.....	46

6.3. Brute-Force Search	46
6.3.1. Basic Algorithm	47
6.4. Heuristic	48
6.4.1. Definition and Motivation	48
6.4.2. Genetic Algorithm	49
6.4.2.1. Methodology – Optimization Problems	50
6.4.2.2. Initialization	51
6.4.2.3. Selection	52
6.4.2.4. Roulette Wheel Selection	52
6.4.2.5. Genetic Operators	54
6.4.2.6. Termination	57
6.4.3. Hill Climbing Algorithm	58
7. Optimization	60
7.2. Optimization of the 1 st Intersection	60
7.3. Optimization of the 2 nd Intersection	66
7.4. Optimization of the 3 rd Intersection	72
8. Synchronization	78
8.1. Study Case	83
9. Conclusion	87
Appendix A	88

Appendix B	95
------------------	----

Alireza Alipourfaz

The details of all the junctions are shown on the table 1.

Intersection	NB	SB	EB	WB
1	Viale Ronchi	Largo Irpinia	via Prenestina	via Prenestina
2	Via Dignano d'Istria	Largo Irpinia	via Prenestina	via Prenestina
3	Via Olevano Romano	-	via Prenestina	via Prenestina

Table 2.1: Details of each intersection

Table 2 presents the distance between each intersection and total distance of 440 m.

Intersection	Distance (m)
1 - 2	120
2 - 3	320
1 - 3	440

Table 2.2: Distance between consecutive intersections

3. Data Collection

Several forms of data exist when it comes to traffic monitoring. Common input data, such as vehicle counts, classifications, and turning movements are often collected to structure the project.

A peak is a part of the day during which traffic congestion on intersections and crowding on public transport is at its highest. Normally, this happens twice every weekday, once in the morning and once in the afternoon-evening, the times during which the most people commute.

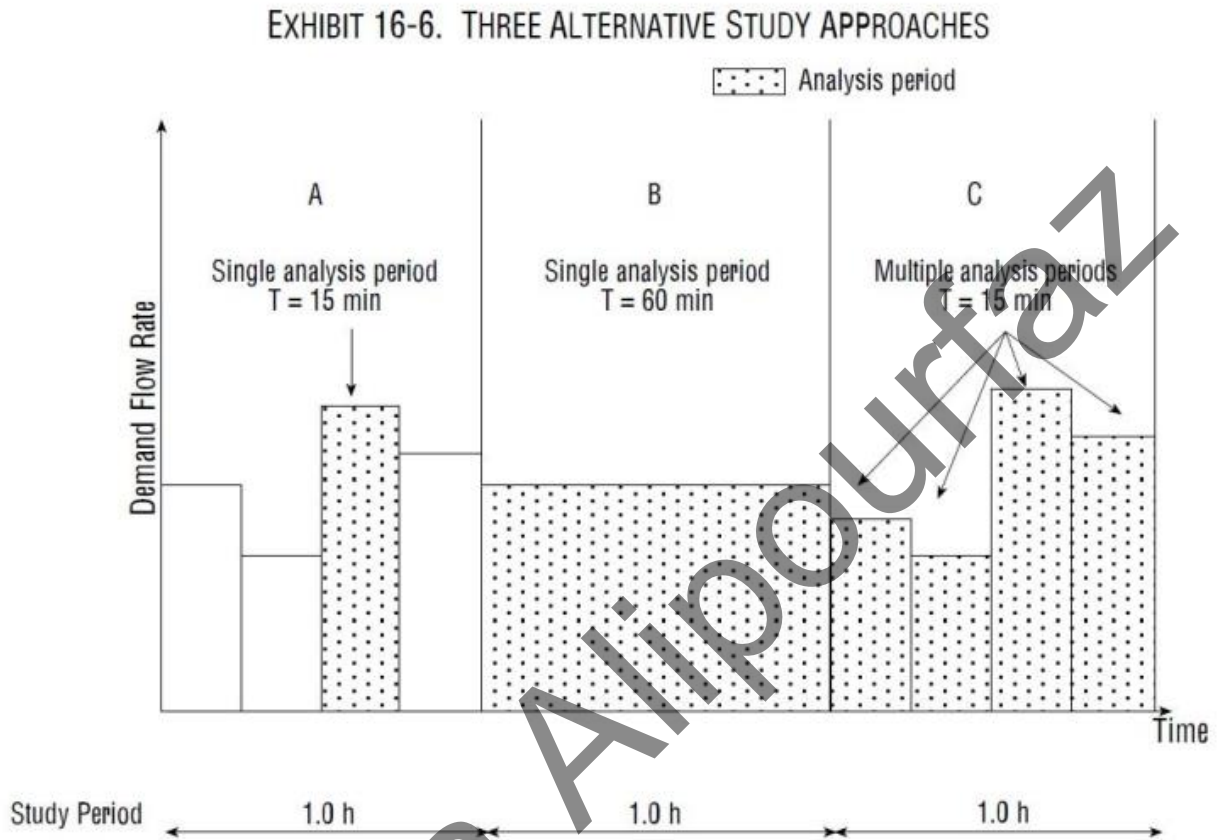
The peak hour factor is used in HCM capacity and level of service analysis to account for the variation in traffic volumes during the peak hour.

Traffic engineers focus on the peak-hour traffic volume in evaluating capacity and other parameters because it represents the most critical time. And, as any motorist who travels during the morning or evening rush hours knows, it's the period during which traffic volume is at its highest. The analysis of level of service is based on peak rates of flow occurring within the peak hour because substantial short-term fluctuations typically occur during an hour. Widespread practice is to use a peak 15-minute rate of flow. Flow rates are usually expressed in vehicles per hour, not vehicles per 15 minutes. The relationship between the peak 15-minute flow rate and the full hourly volume is given by the peak-hour factor (PHF). In this project manual method has been used to collect data, Manual—Refers to visually observing number, classification, vehicle occupancy, turning movement counts, or direction of traffic. Due to COVID-19 virus, in this semester we could not collect data in the street. So, we are using the data that presented by professor Gaetano Fusco.

4. Methodology

Two major analytic steps are performed in the volume adjustment module. Movement volumes are adjusted to flow rates for each desired period of analysis. If necessary, and lane groups for analysis are established. Figure 2 demonstrates three alternative ways in which an analyst might proceed for a given study. Also, other alternatives exist. Approach A is the one that has traditionally been used in the HCM, and also in this project. The length of the period being analyzed is only 15 min, and the analysis period (T), therefore, is 15 min or 0.25 h. In this case, either a peak 15-min volume is available or one is derived from an hourly volume by use of a PHF. A difficulty with considering only one 15 min period is that a queue may be left at the end of the analysis period because of demand in excess of capacity. In such cases it is possible that the queue carried over to the next

period will result in delay to vehicles that arrive in that period beyond that which would have resulted had there not been a queue carryover.



All the analysis in this project have been based on the HCM, following 5 main steps reported in Figure 3.

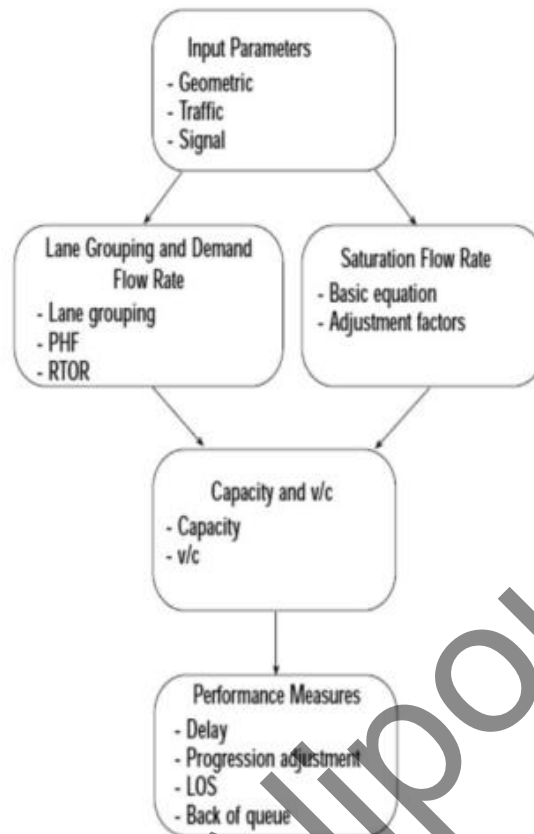


Figure 3.2: Signalized intersection methodology (Font: HCM chap.16)

4.1. Input Parameters

They are defined:

- ✓ Geometric characteristics of the roads approaching the intersection: intersection geometry is generally presented in diagrammatic form and must include all of the relevant information, including: approach grades, the number and width of lanes, and parking conditions;
- ✓ Traffic volumes for the intersection must be specified for each movement on each approach; it is included: number of movements [vph] for every approach and their division according to direction (LT, TH, RT), base saturation flow S_0 , Peak Hour Factor PHF, percentage of heavy vehicles, Arrival Type.

✓ Signalization conditions include a phase diagram illustrating the phase plan, cycle length, green times, and change-and-clearance intervals.

4.2. Lane Grouping and Demand Flow Rate

The analysis has been done disaggregating each approach in the corresponding lane groups.










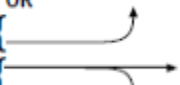



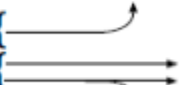
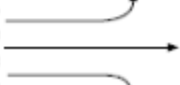
Number of Lanes	Movements by Lanes	Number of Possible Lane Groups
1	LT + TH + RT 	①  (Single-lane approach)
2	EXC LT  TH + RT 	②  
2	LT + TH  TH + RT 	①  OR ② 
3	EXC LT  TH  TH + RT 	②  OR ③ 

Figure 4.1: Typical lane groups for analysis (Font: HCM chap.16)

In Figure 4, the typical lane groups for analysis are reported.

As said before, the flows have been counted for 15 min during the peak hour, so the

approach A in figure 2 has been used to obtain the flow rate during peak – 15 min period.

$$V_p = \frac{V}{PHF}$$

Where:

V_p = flow rate during peak -15- min period (veh/h)

V = hourly volume (veh/h)

PHF = peak-hour factor

In this project the peak hour factor has been assumed equal to 0.95, standard value for artery into the city, considering that traffic conditions don't change substantially during the peak hour.

Vehicles have been counted dividing them into categories, which comprise also heavy vehicles and buses, and into direction movements.

4.2. Saturation Flow Rate

The saturation flow rate can be defined as the flow in vehicles per hour that can be accommodated by the lane group assuming that the green phase was displayed 100 percent of the time and it can be computed according to the following equation:

$$S = S_o \cdot N \cdot f_w \cdot f_{HV} \cdot f_g \cdot f_p \cdot f_{bb} \cdot f_a \cdot f_{LU} \cdot f_{LT} \cdot f_{RT} \cdot f_{Lpb} \cdot f_{Rpb}$$

S = saturation flow rate for subject lane group, expressed as a total for all lanes in lane group
(veh/h);

S_o = base saturation flow rate per lane (pc/h/ln); it has been set at 2100 vph/lane

N = number of lanes in lane group;

f_w = adjustment factor for lane width;

f_{HV} = adjustment factor for heavy vehicles in traffic stream;

f_g = adjustment factor for approach grade;

f_p = adjustment factor for existence of a parking lane and parking activity adjacent to lane group;

f_{bb} = adjustment factor for blocking effect of local buses that stop within intersection area;

f_a = adjustment factor for area type;

f_{LU} = adjustment factor for lane utilization (Exhibit 10-23);

f_{LV} = adjustment factor for left turns in lane group;

f_{RT} = adjustment factor for right turns in lane group;

f_{LPB} = pedestrian adjustment factor for left-turn movements;

f_{RPB} = pedestrian-bicycle adjustment factor for right-turn movements.

4.4. Capacity

The capacity of each lane group is the maximum number of vehicles that can be served according to the signal regulation:

$$c_i = s_i \frac{g_i}{C}$$

Where:

c_i = capacity of lane group i (veh/h)

s_i = saturation flow rate for lane group i (veh/h)

g_i/C = effective green ratio for lane group i

4.5. v/c Ratio

The ratio of flow rate to capacity (v/c), often called the volume to capacity ratio, is given by the symbol X in intersection analysis. It is typically referred to as degree of saturation. For a given lane group i , X_i is computed using the Equation below:

$$X_i = \left(\frac{v}{c}\right)_i = \frac{v_i}{s_i \left(\frac{g_i}{C}\right)} = \frac{v_i C}{s_i g_i}$$

4.6. Delay

One way to check an existing or planned signal timing scheme is to calculate the delay experienced by those who are using, or who will use, the intersection. The delay experienced by the average vehicle can be directly related to a level of service (LOS). The LOS categories, which are listed below, contain information about the progression of traffic under the delay conditions that they represent. This allows to visualize and understand the traffic flow conditions surrounding an intersection, even though the intersection might still be on the drawing board.

$$d = d_1(PF) + d_2 + d_3$$

Where:

d = control delay per vehicle (s/Veh);

d_1 = uniform control delay assuming uniform arrivals (s/veh);

$$d_1 = \frac{0.5C(1 - \frac{g}{C})^2}{1 - \left[\min(1, X) \frac{g}{C}\right]}$$

C = Cycle length (s)

g = effective green time for lane group (s)

$X = v/c$ ratio or degree of saturation for lane group

PF = uniform delay progression adjustment factor, which accounts for effects of signal progression

d_2 = incremental delay to account for effect of random arrivals and oversaturation queues, adjusted for duration of analysis period and type of signal control; this delay component assumes that there is no initial queue for lane group at the start of the analysis period (s/veh); and

$$d_2 = 900T \left[(X - 1) + \sqrt{(X - 1)^2 + \frac{8kIX}{cT}} \right]$$

T = duration of analysis period (h)

k = incremental delay factor that is dependent on controller setting; (here, the signals are pre-timed and k = 0.5)

l = upstream filtering/metering adjustment factor;

(value of 1 is used because these are isolated intersections)

c = lane group capacity (veh/h)

X = v/c ratio or degree of saturation for lane group

d_3 = initial queue delay, which accounts for delay to all vehicles in analysis period due to initial queue at the start of the analysis period (s/veh) (considered zero in this project: no queue at the beginning of the observation period). Thus, the delay for an approach is computed using the equation below:

$$d_A = \frac{\sum d_i v_i}{\sum v_i}$$

Where:

d_A = delay for Approach A (s/veh),

d_i = delay for lane group i (on Approach A) (s/veh)

v_i = adjusted flow for lane group i (veh/h)

Control delays on the approaches can be further aggregated using equation above to provide the average control delay for the intersection:

$$d_I = \frac{\sum d_A v_A}{\sum v_A}$$

Where:

d_I = delay per vehicle for intersection (s/veh)

d_A = delay for Approach A (s/veh)

v_A = adjusted flow for Approach A (veh/h)

4.7. Level of Service (LOS)

The LOS is related to the average control delay per vehicle. The appropriate LOS can be determined by estimating the delay for each lane group and aggregated for each approach and the intersection. LOS is a measure of the delay incurred by vehicles at a signalized intersection. In some cases, delay will be high even when v/C ratios are low. In these situations, poor progression or an inappropriately long cycle length, or both, is generally the cause. So, an intersection can have an unacceptably high delays without there being a capacity problem. When v/c approaches or exceeds 1.0, it is possible that delay will remain at acceptable level. This situation can occur, especially if the time over which high v/c level occur is short. It can also occur if the analysis is for only a single period and there is queue carryover. By having multi-period analysis, we can gain a picture of delay close to the reality. The criteria of the LOS with respect to the control delay per vehicle are shown in the Table 3.

LOS	Control Delay per vehicle(s/veh)
A	≤ 10
B	$> 10 - 20$
C	$> 20 - 35$
D	$> 35 - 55$
E	$> 55 - 80$
F	> 80

Table 4.1: Level of Service (LOS) criteria for signalized intersections (Font: HCM chap. 16)

5. Junctions' analysis: Current situation

In this section each intersection has been analyzed based on the HCM method, as a single junction, such that the impact of the close intersections does not affect the analyzed one.

5.1.1 Junction 1: Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina

This Junction is Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina which is composed by 3 approaches has a signal setting composed by Three phases.

Figure 5.2 shows a geometrical representation of the junction, with the characteristics reported in Table 5.1.

Approach	Number of lanes	Lane width (m)	Grade %
SB	1	3.7	0
EB	2	3.5	0
WB	3	3.5	0

Table 5.1: Geometrics characteristics of 1st Intersection

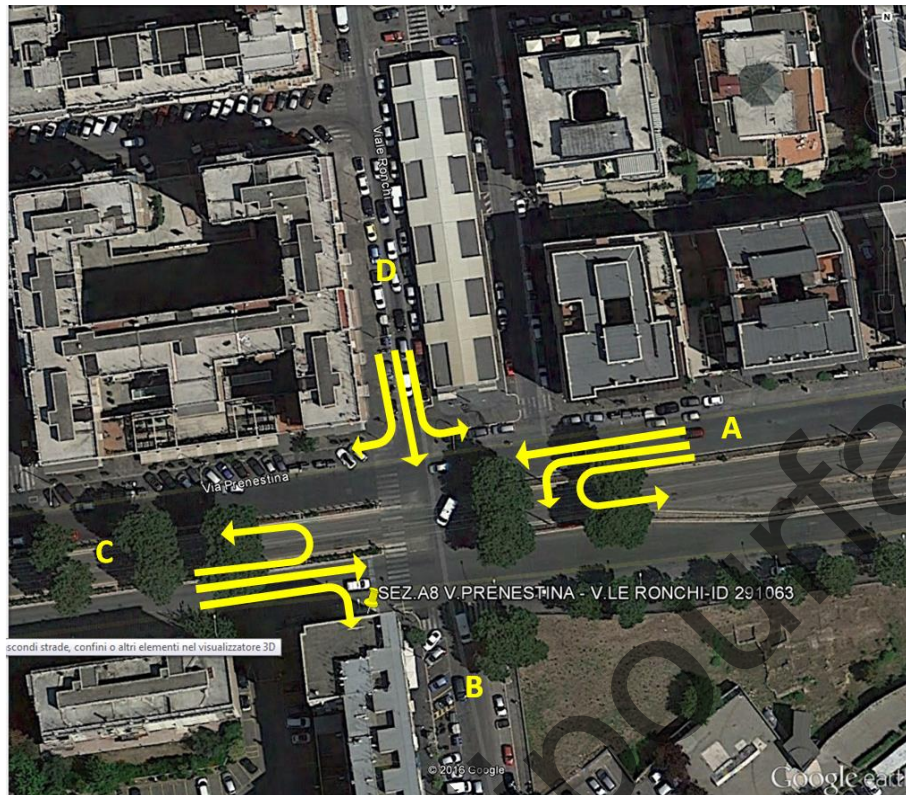


Figure 5.1: Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina

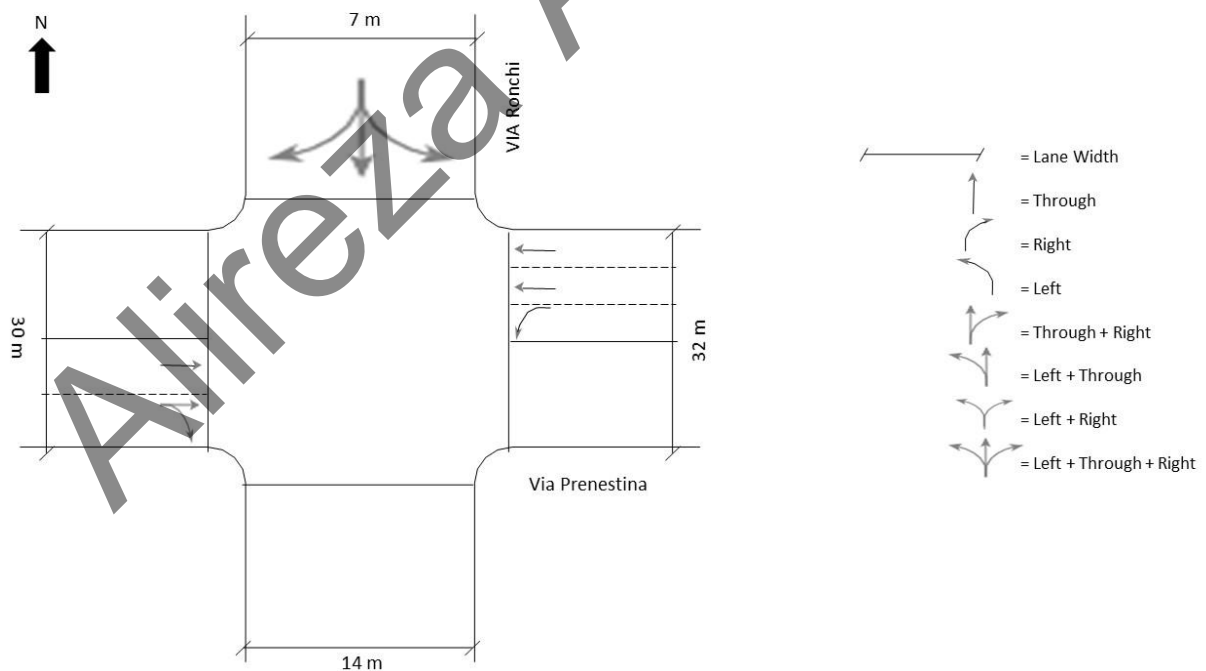


Figure 5.2 Geometric representation of the 1st intersection

Table 5.2 shows the bus lines stopping at the intersection.

Bus Number	Bus Line Stopping	Road
113	Prenestina/Irpinia	EB & WB
314	Prenestina/Irpinia	EB & WB
501	Prenestina/Irpinia	EB & WB
n5	Prenestina/Irpinia	EB & WB
n075	Prenestina/Irpinia	EB & WB
n543	Prenestina/Irpinia	EB & WB

Table 5.2: Bus lines stopping at 1st intersection

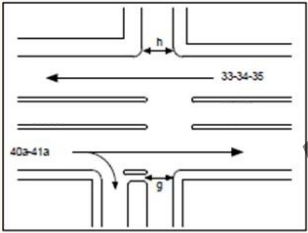
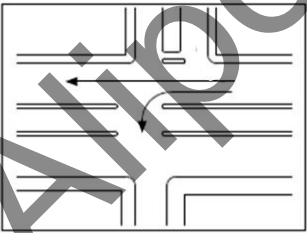
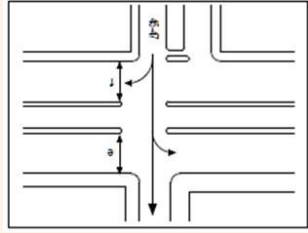
Phases	1	2	3
Diagram			
Green (s)	43	33	38
Yellow (s)	4	4	4
Red (s)	85	97	90
Cycle (s)	132		

Table 5.3: Phase planning of 1st intersection

INPUT WORKSHEET													
GENERAL INFORMATION						SITE INFORMATION							
ANALYST	Alireza Alipourfaz					INTERSECTION		Viale Ronchi - Largo Irpinia - via Prenestina					
AGENCY OR COMPANY						AREA TYPE		CBD		OTHERS			
DATE PERFORMED	5/7/2020					JURISDICTION							
ANALYSIS TIME PERIOD	8:00 - 8:15					ANALYSIS YEAR		2020					
VOLUME AND TIME INPUT													
	EB			WB			NB			SB			
	LT	TH	RT^1	LT	TH	RT^1	LT	TH	RT^1	LT	TH	RT^1	
Volume, V (veh/h)	0	740	84	280	1504	0				64	232	36	
Heavy vehicles, % HV	15			9.1						6			
Peak-hour factor, PHF	0.9			0.9						0.9			
Pretimed (P) or actuated (A)	P			P						P			
Start-up lost time, I1 (s)	2.5			2.5						2.5			
Clearance lost time (s)	2.3			2.3						3.6			
Total lost time (s)	4.8			4.8						6.1			
Arrival type, AT	3			3						3			
Approach pedestrian volume, 2 vped (p/h)	50			50						50			
Approach bicycle volume, 2 vbic (bicycles/h)	20			20						20			
Parking (Y or N)	Y			Y						Y			
Parking maneuvers, Nm (maneuvers/h)	8			8						12			
Bus stopping, NB (buses/h)	6			10						N			
Min. timing for pedestrians, 3 Gp (s)	15.2			15.2						18.0			
SIGNAL PHASING PLAN													
DIAGRAM	Ø1	Ø2	Ø3	Ø4	Ø5	Ø6	Ø7	Ø8					
Timing	G= 43 Y= 4	G= 33 Y= 4	G= 38 Y= 4		G= Y=	G= Y=	G= Y=	G= Y=					
Permitted Turns	-----		Permitted Turns -----	Pedestrian		Cycle Length, C= 132 s							
NOTES													
1. RT volumes, as shown, exclude RTOR.													
2. Approach pedestrian and bicycle volumes are those that conflict with right turns from the subject approach.													
3. Refer to Equation 16-2.													

Table 5.4: Volume and timing input of 1st intersection

In Table 5.4 volume and timing input are summarized. The traffic signal is pretimed. The start-up lost time is assumed to be 2.5 s, which is a feasible value for urban area during the peak hour. Arrival type is assumed 3 corresponding to random arrivals.

Table 5.3 shows the phase planning of the intersection. The table shows the timing of the signal. The cycle length of this junction is 132 seconds which consist of 3 signal phases. The first phase is for EB and WB approach, the second phase is for EB and WB approach, and the third phase is for the SB approach. The first phases have 43 seconds of green, 4 seconds of yellow, and 85 seconds of red. The second phase has 33 seconds of green, 4 seconds of yellow, and 97 seconds of red. The thirds phase has 38 seconds of green, 4 seconds of yellow, and 90 seconds of red. There are 6 seconds of total red which guarantee the clearance of the conflicting area.

VOLUME ADJUSTMENT AND SATURATION FLOW RATE WORKSHEET												
General Information												
Project Description: Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina												
Volume Adjustment												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Volume, V (veh/h)	0	740	84	280	1504	0	0	0	0	64	232	36
Peak-hour factor, PHF	0.9			0.9						0.9		
Adjusted flow rate, $v_p = V/PHF$ (veh/h)	0	822	93	311	1671	0				71	258	40
Lane Group												
Adjusted flow rate in lane group, v (veh/h)	915			311	1671					369		
Proportion of LT or RT (P_LT, P_RT)		-	0.102	1	-					0.192	-	0.108
Saturation flow rate												
base saturation flow, S_0 (pc/h/ln)	2100			2100	2100					2100		
number of lanes, N	2			1	2					1		
Lane width adjustment factor, f_W	0.989			0.989	0.989					1.011		
Heavy-vehicle adjustment factor, f_{HV}	0.870			0.917	0.917					0.943		
Grade adjustment factor, f_g	1			1	1					1		
Parking adjustment factor, f_p	0.949			1	0.950					0.948		
Bus blockage adjustment factor, f_{bb}	0.988			1	0.904					1		
Area type adjustment factor, f_a	1			1	1					1		
Lane Utilization adjustment factor, f_{LU}	0.952			1	0.952					1		
Left-Turn adjustment factor, f_{LT}	1			0.95	1					0.990		
Right-Turn adjustment factor, f_{RT}	0.985			1	1					0.984		
Left-Turn ped/bike adjustment factor, f_{Rpb}	1			0.783	1					0.979		
right-Turn ped/bike adjustment factor, f_{Rpb}	0.975			1	1					0.986		
Adjusted saturation flow, s (veh/h) $s = s_0 * N * f_w * f_{hv} * f_g * f_{fb} * f_a * f_{LU} * f_{LT} * f_{RT} * f_{Lpb} * f_{Rpb}$	3098			1417	3114					1807		

Table 5.5: Volume adjustment and saturation flow rate of the 1st intersection

In Table 5.5 the capacity analysis of the intersection has been reported. The analysis considered each lane group as a single entity.

The effective green time is the time useful for crossing the conflicting area, so green plus yellow minus the lost time for the phase. It is necessary to determine the capacity of the lane group. Then it's possible to define the volume to capacity ratio, which can indicate, if it's higher than 1, that there is an excess of demand over capacity.

SUPPLEMENTAL WORKSHEET FOR PEDESTRIAN-BICYCLE EFFECTS ON PERMITTED LEFT TURNS AND RIGHT TURNS												
General Information												
Project Description: Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina												
Permitted Left Turns												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Effective pedestrian green time, 1,2 gp (s)						15.2						18
Conflicting pedestrian volume, 1 vped (p/h)						50						50
vpedg = vped (C/gp)						434						367
OCCpedg = vpedg/2000 if (vpedg ≤ 1000) or OCCpedg = 0.4 + vpedg/10,000 if (1000 < vpedg ≤ 5000)						0.217						0.1835
Opposing queue clearing green, 3,4 gq (s)						0						0
Effective pedestrian green consumed by opposing vehicle queue, gq/gp; if gq ≥ gp then flpb = 1.0						0						0
OCCpedu = OCCpedg [1 - 0.5(gq/gp)]						0.217						0.1835
Opposing flow rate, 3 vo (veh/h)						0						0
OCCr = OCCpedu [e ^{-(5/3600)vo}]						0.217						0.1835
Number of cross-street receiving lanes, 1 Nrec						1						2
Number of turning lanes, 1 Nturn						1						1
ApbT = 1 - OCCr if Nrec = Nturn ApbT = 1 - 0.6(OCCr) if Nrec > Nturn						0.783						0.890
Proportion of left turns, 5 PLT						1.000						0.192
Proportion of left turns using protected phase, 6 PLTA						0						0
flpb = 1.0 - PLT(1 - ApbT)(1 - PLTA)						0.783						0.979
Permitted Right Turns												
	EB			WB			NB			SB		
Effective pedestrian green time, 1,2 gp (s)						15.2						18
Conflicting pedestrian volume, 1 vped (p/h)						50						50
Conflicting bicycle volume, 1,7 vbic (bicycles/h)						20						20
vpedg = vped(C/gp)						421						356
OCCpedg = vpedg/2000 if (vpedg ≤ 1000), or OCCpedg = 0.4 + vpedg/10,000 if (1000 < vpedg ≤ 5000)						0.211						0.178
Effective green, 1 g (s)						42.2						35.9
vbicg = vbic(C/g)						63						74
OCCbicg = 0.02 + vbicg/2700						0.043						0.047
OCCr = OCCpedg + OCCbicg - (OCCpedg)(OCCbicg)						0.245						0.217
Number of cross-street receiving lanes, 1 Nrec						1						2
Number of turning lanes, 1 Nturn						1						1
ApbT = 1 - OCCr if Nrec = Nturn ApbT = 1 - 0.6(OCCr) if Nrec > Nturn						0.755						0.870
Proportion of right turns, 5 PRT						0.102						0.108
Proportion of right turns using protected phase, 8 PRTA						0						0
fRpb = 1.0 - PRT(1 - ApbT)(1 - PRTA)						0.975						0.986

Table 5.6: Supplemental Table for Pedestrian-Bicycle effects on Permitted Right Turns of the 1st intersection





CAPACITY AND LOS WORKSHEET												
General Information												
Project Description:Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina												
Capacity Analysis												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Phase number	1			2	1/2					3		
Phase type	P			P	P					P		
Lane Group												
Adjusted flow rate, V (veh/h)	915			311	1671					369		
Saturation flow rate, S (veh/h)	3098			1417	3114					1807		
Lost time t_L (s), $t_L = I_1 + Y - \theta$	4.8			4.8	4.8					6.1		
Effective green timeg (s), $g = G + Y - t_L$	42.2			32.2	84					35.9		
Green ratio , g/C	0.320			0.244	0.636					0.272		
Lane group capacity, $c=5*(g/C)$, (veh/h)	991			346	1981					492		
V/c ratio , X	0.924			0.900	0.843					0.751		
Flow ratio, V/S	0.295			0.219	0.537					0.204		
Critical Lane group/phase (√)					√					√		
Sum of flow ratios for critical lane groups, Y_c $Y_c = \sum (\text{critical lane groups, v/s})$	0.741											
Total Lost time per cycle, L (s)	15.7											
Critical flow rate to capacity ratio, X_c $X_c = (Y_c)/(C)/(C - L)$	0.841											

Table 5.7: Capacity analysis of the 1st intersection

In Table 5.8 the computation of the delay and the evaluation of LOS (Level of Service) is reported. In the analysis period there is no initial queue, so $d_3 = 0$. Progression factor (PF) is based on the arrival type.

The first calculation is the delay (and consequently the LOS) of each lane group.

The level of service of the whole intersection is D.





Lane Group Capacity, Control Delay, and LOS Determination						
	EB	WB			NB	SB
Lane Group						
Adjusted flow rate, V (veh/h)	915	311	1671			369
V/c ratio , X=V/c	0.924	0.900	0.843			0.751
Total Green ration , g/C	0.32	0.244	0.636			0.272
Uniform delay, $d_1 = \frac{0.50 C [1 - (g/C)^2]}{1 - [\min(1, X)g/C]}$ (s/veh)	43	48	19			44
Incremental Delay Calibration, K	0.5	0.5	0.5			0.5
Incremental delay, ⁴ d_2 $d_2 = 900T[(X - 1) + \sqrt{(X - 1)^2 + \frac{8kKX}{cT}}]$ (s/veh)	15.3	28.6	4.6			10.1
Initial queue Delay, d3 (s/veh)	0	0	0			0
Uniform Delay, d1 (s/veh)						
Progression Adjustment Factor, PF	1	1	1			1
Delay, $d = d_1(PF) + d_2 + d_3$ (s/veh)	58.3	76.6	23.6			54.1
LOS by lane group	E	E	C			D
Delay by approach, $d_A = \frac{\sum(d)(v)}{\sum V}$ (s/veh)	58.3	31.9				54.1
LOS by Approach	E	C				D
Approach flow rate, V_A (veh/h)	915	1982				369
Intersection delay, $d_I = \frac{\sum(d_A)(V_A)}{\sum V_A}$ (s/veh)	41.8	Intersection LOS				D

 Table 5.8: Delay and Level of Service of the 1st intersection

5.2.2 Junction 2: Via Dignano d'Istria – Largo

Irpinia - via Prenestina - via Prenestina

This Junction is Via Dignano d'Istria - Largo Irpinia - via Prenestina - via Prenestina which is composed by 3 approaches has a signal setting composed by Two phases.

Figure 5.4 shows a geometrical representation of the junction, with the characteristics reported in Table 5.9.

Approach	Number of lanes	Lane width	Grade %
NB	3	3.5	0
EB	2	3.5	0
WB	2	3.5	0

Table 5.9: Geometrics characteristics of 2nd Intersection

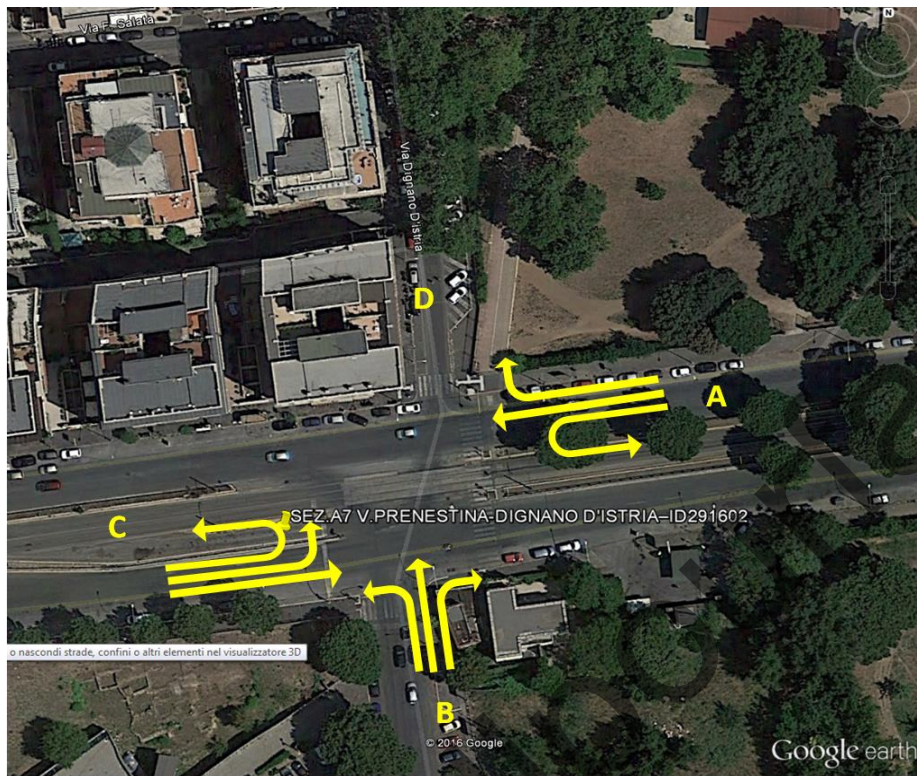


Figure 5.3: Via Dignano d'Istria - Largo Irpinia - via Prenestina - via Prenestina

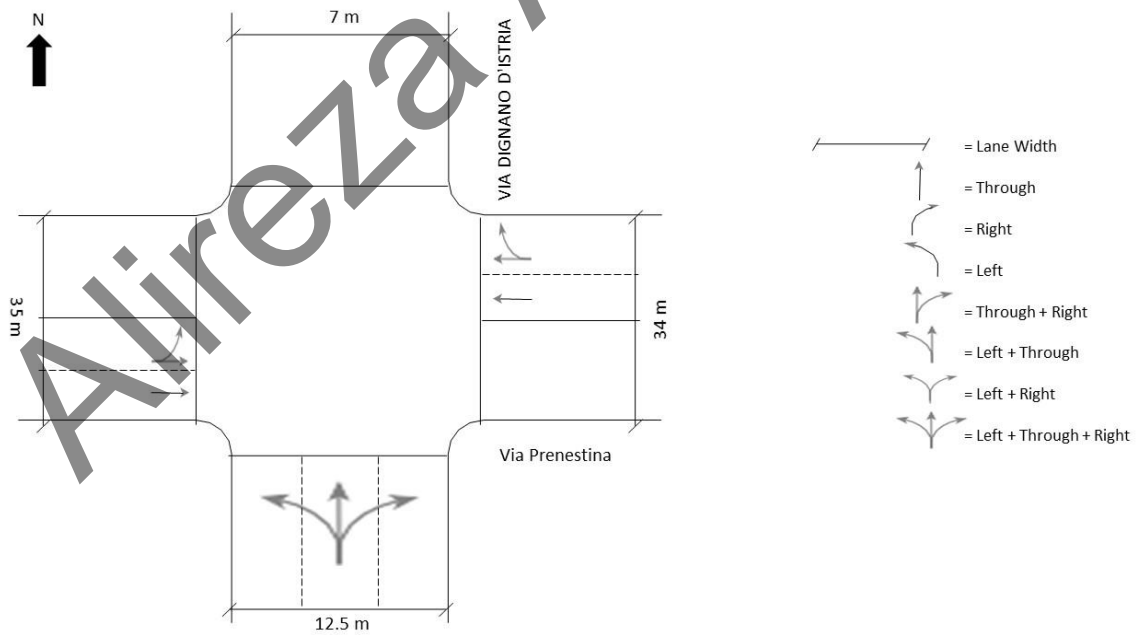


Figure 5.4: Geometric representation of the 2nd intersection

Table 5.28 shows the bus lines stopping at the intersection.

Bus Number	Bus Line Stopping	Road
113	Prenestina/Irpinia	EB & WB
314	Prenestina/Irpinia	EB & WB
501	Prenestina/Irpinia	EB & WB
n5	Prenestina/Irpinia	EB & WB
n075	Prenestina/Irpinia	EB & WB
n543	Prenestina/Irpinia	EB & WB

Table 5.10: Bus lines stopping at 2nd intersection

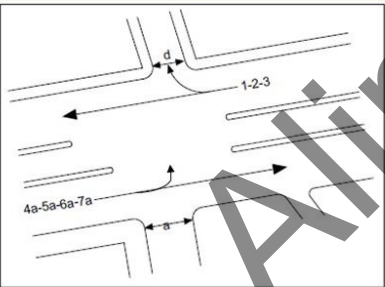
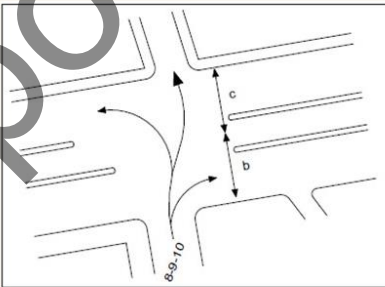
Phases	1	2
Diagram		
Green (s)	76	44
Yellow (s)	4	4
Red (s)	50	82
Cycle (s)	132	

Table 5.11: Phase plane of 2nd intersection

INPUT WORKSHEET																
GENERAL INFORMATION						SITE INFORMATION										
ANALYST	Alireza Alipourfaz					INTERSECTION		Via Dignano d'Istria - Largo Irpinia - via Prenestina - via Prenestina								
AGENCY OR COMPANY						AREA TYPE		CBD				OTHERS				
DATE PERFORMED	5/7/2020					JURISDICTION										
ANALYSIS TIME PERIOD	8:00 - 8:15					ANALYSIS YEAR		2020								
VOLUME AND TIME INPUT																
	EB			WB			NB			SB						
	LT	TH	RT^1	LT	TH	RT^1	LT	TH	RT^1	LT	TH	RT^1				
Volume, V (veh/h)	4	736			1804	164	140	328	128							
Heavy vehicles, % HV	5.3			3.8			4.4									
Peak-hour factor, PHF	0.9			0.9			0.9									
Pretimed (P) or actuated (A)	P			P			P									
Start-up lost time, I1 (s)	2.5			2.5			2.5									
Clearance lost time (s)	2.8			2.8			3.1									
Total lost time (s)	5.3			5.3			5.6									
Arrival type, AT	3			3			3									
Approach pedestrian volume, 2 vped (p/h)	50			50			50									
Approach bicycle volume, 2 vbic (bicycles/h)	20			20			20									
Parking (Y or N)	Y			Y			Y									
Parking maneuvers, Nm (maneuvers/h)	8			8			3									
Bus stopping, NB (buses/h)	10			10			N									
Min. timing for pedestrians, 3 Gp (s)	9.7			9.7			13.1									
SIGNAL PHASING PLAN																
DIAGRAM	Ø1		Ø2		Ø3		Ø4		Ø5		Ø6		Ø7		Ø8	
Timing	G=	76	G=	44	G=		G=		G=		G=		G=		G=	
	Y=	4	Y=	4	Y=		Y=		Y=		Y=		Y=		Y=	
Permitted Turns				Permitted Turns				Pedestrian				Cycle Length, C= 132 s				

Table 5.12: Volume and timing input of 2nd intersection

In Table 5.12 volume and timing input are summarized. The traffic signal is pretimed. The start-up lost time is assumed to be 2.5 s, which is a feasible value for urban area during the peak hour. Arrival type is assumed 3 corresponding to random arrivals.

Table 5.11 shows the phase planning of the intersection. There is no overlap. The table shows the timing of the signal. The cycle length of this junction is 132 seconds which consist of 2 signal phases. The first phase is for EB and WB approach. The second phase is for the NB approach. The first phases have 76 seconds of green, 4 seconds of yellow, and 50 seconds of red. The second phase has 44 seconds of green, 4 seconds of yellow, and 82 seconds of red. There are 4 seconds of total red which guarantee the clearance of the conflicting area.

VOLUME ADJUSTMENT AND SATURATION FLOW RATE WORKSHEET												
General Information												
Project Description: Via Dignano d'Istria - Largo Irpinia - via Prenestina - via Prenestina												
Volume Adjustment												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Volume, V (veh/h)	4	736	0	0	1804	164	140	328	128	0	0	0
Peak-hour factor, PHF	0.9			0.9			0.9			0		
Adjusted flow rate, $v_p = V / PHF$ (veh/h)	4	818	0	0	2004.4	182.2	155.6	364.4	142.2			
Lane Group												
Adjusted flow rate in lane group, v (veh/h)	822			2187			662					
Proportion of LT or RT (P_LT, P_RT)	0.005	-			-	0.083	0.235	-	0.215		-	
Saturation flow rate												
base saturation flow, S_0 (pc/h/ln)	2100			2100			2100					
number of lanes, N	2			2			3					
Lane width adjustment factor, f_W	0.989			0.989			0.989					
Heavy-vehicle adjustment factor, f_{HV}	0.950			0.963			0.958					
Grade adjustment factor, f_g	1			1			1					
Parking adjustment factor, f_p	0.949			0.949			0.950					
Bus blockage adjustment factor, f_{bb}	0.980			0.980			1.000					
Area type adjustment factor, f_a	1			1			1					
Lane Utilization adjustment factor, f_{LU}	0.952			0.952			0.908					
Left-Turn adjustment afctor, f_{LT}	0.836			1			0.988					
Right-Turn adjustment afctor, f_{RT}	1			0.988			0.968					
Left-Turn ped/bike adjustment afctor, f_{Rpb}	1			1			0.965					
right-Turn ped/bike adjustment afctor, f_{Rpb}	1			0.982			0.964					
Adjusted saturation flow, s (veh/h) $s = s_0 * N * f_w * f_{hv} * f_g * f_p * f_{bb} * f_a * f_{LU} * f_{LT} * f_{RT} * f_{Lpb} * f_{Rpb}$	2921			3454			4652					

Table 5.13: Volume adjustment and saturation flow rate of the 2nd intersection

In Table 5.13 the capacity analysis of the intersection has been reported. The analysis considered each lane group as a single entity.

The effective green time is the time useful for crossing the conflicting area, so green plus yellow minus the lost time for the phase. It is necessary to determine the capacity of the lane group. Then it's possible to define the volume to capacity ratio, which can indicate, if it's higher than 1, that there is an excess of demand over capacity.

SUPPLEMENTAL WORKSHEET FOR PEDESTRIAN-BICYCLE EFFECTS ON PERMITTED LEFT TURNS AND RIGHT TURNS												
General Information												
Project Description:Via Dignano d'Istria - Largo Irpinia - via Prenestina - via Prenestina												
Permitted Left Turns												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Effective pedestrian green time,1,2 gp (s)	9.7						13.1					
Conflicting pedestrian volume,1 vped (p/h)	50						50					
vpedg = vped (C/gp)	660						489					
OCCpedg = vpedg/2000 if (vpedg ≤ 1000) or OCCpedg = 0.4 + vpedg/10,000 if (1000 < vpedg ≤ 5000)	0.330						0.245					
Opposing queue clearing green,3,4 gq (s)	73.2						0					
Effective pedestrian green consumed by opposing vehicle queue, gq/gp; if gq ≥ gp then flpb = 1.0	1						0					
OCCpedu = OCCpedg [1 – 0.5(gq/gp)]							0.245					
Opposing flow rate,3 vo (veh/h)							0					
OCCr = OCCpedu [e [^] (–(5/3600)vo)]							0.245					
Number of cross-street receiving lanes,1 Nrec							2					
Number of turning lanes,1 Nturn							1					
ApbT = 1 – OCCr if Nrec = Nturn ApbT = 1 – 0.6(OCCr) if Nrec > Nturn							0.853					
Proportion of left turns,5 PLT							0.235					
Proportion of left turns using protected phase,6 PLTA							0					
flpb = 1.0 – PLT(1 – ApbT)(1 – PLTA)	1						0.965					
Permitted Right Turns												
	EB			WB			NB			SB		
Effective pedestrian green time,1,2 gp (s)				9.7			13.1					
Conflicting pedestrian volume,1 vped (p/h)				50			50					
Conflicting bicycle volume,1,7 vbic (bicycles/h)				20			20					
vpedg = vped(C/gp)				660			489					
OCCpedg = vpedg/2000 if (vpedg ≤ 1000), or OCCpedg = 0.4 + vpedg/10,000 if (1000 < vpedg ≤ 5000)				0.330			0.245					
Effective green,1 g (s)				74.7			42.4					
vbicg = vbic(C/g)				34			60					
OCCbicg = 0.02 + vbicg/2700				0.033			0.042					
OCCr = OCCpedg + OCCbicg – (OCCpedg)(OCCbicg)				0.352			0.277					
Number of cross-street receiving lanes,1 Nrec				2			3					
Number of turning lanes,1 Nturn				1			1					
ApbT = 1 – OCCr if Nrec = Nturn ApbT = 1 – 0.6(OCCr) if Nrec > Nturn				0.789			0.834					
Proportion of right turns,5 PRT				0.083			0.215					
Proportion of right turns using protected phase,8 PRTA				0			0					
fRpb = 1.0 – PRT(1 – ApbT)(1 – PRTA)				0.982			0.964					

Table 5.14: Supplemental Table for Pedestrian-Bicycle effects on Permitted Right Turns of the 2nd intersection

SUPPLEMENTAL WORKSHEET FOR PERMITTED LEFT TURNS OPPOSED BY MULTILANE APPROACH												
General Information												
Project Description:Via Dignano d'Istria - Largo Irpinia - via Prenestina - via Prenestina												
Input												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Cycle length, C (s)	132											
Total actual green time for LT lane group,1 G (s)	76											
Effective permitted green time for LT lane group,1 g (s)	75											
Opposing effective green time, go (s)	75											
Number of lanes in LT lane group,2 N	2											
Number of lanes in opposing approach, No	2											
Adjusted LT flow rate, vLT (veh/h)	4											
Proportion of LT volume in LT lane group,3 PLT	0.005											
Adjusted flow rate for opposing approach, vo (veh/h)	2004											
Lost time for LT lane group, tL	5.3											
Computation												
	EB			WB			NB			SB		
LT volume per cycle, LTC = vLT.C/3600	0.147											
Opposing lane utilization factor, fLUo (refer to Volume Adjustment and Saturation Flow Rate Worksheet)	0.952											
Opposing flow per lane, per cycle $v_{olc} = \frac{v_o C}{3600 N_o f_{LUo}} \quad (\text{veh/C/ln})$	39											
$g_l = G[e^{-0.882(LTC^{0.717})}] - t_L \quad g_l \leq g \text{ (except for exclusive left-turn lanes)}^1, 4$	55.5											
Opposing platoon ratio, Rpo(refer to Exhibit 16-11)	1											
Opposing queue ratio, qro = max[1 - Rpo(go/C), 0]	0.432											
$g_q = \frac{v_{olc} q_{ro}}{0.5 - [v_{olc}(1 - q_{ro})/g_o]} - t_L \quad v_{olc}(1 - q_{ro})/g_o \leq 0.49$ (note case-specific parameters) ¹	73.2											
$g_u = g - g_q \text{ if } g_q \geq g_f, \text{ or}$ $g_u = g - g_f \text{ if } g_q < g_f$	1.8											
EL1 (refer to Exhibit C16-3)	10.6											
$P_L = P_{LT} \left[1 + \frac{(N-1)g}{(g_l + g_u/E_{L1} + 4.24)} \right]$ (except with multilane subject approach) ⁵	0.011											
fmin = 2(1 + PL)/g	0.027											
$f_m = [g_f/g] + [g_u/g] \left[\frac{1}{1 + P_L(E_{L1} - 1)} \right], (f_{min} \leq f_m \leq 1.00)$	0.762											
fLT = [fm + 0.91(N - 1)]/N (except for permitted left turns) ⁶	0.836											

Table 5.15: SUPPLEMENTAL WORKSHEET FOR PERMITTED LEFT TURNS OPPOSED BY MULTILANE APPROACH of the 2nd intersection



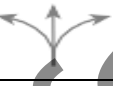
CAPACITY AND LOS WORKSHEET												
General Information												
Project Description:Via Dignano d'Istria - Largo Irpinia - via Prenestina - via Prenestina												
Capacity Analysis												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Phase number	1			1			2					
Phase type	P			P			P					
Lane Group												
Adjusted flow rate, V (veh/h)	822			2187			662					
Saturation flow rate, S (veh/h)	2945			3454			4652					
Lost time t_L (s), $t_L = I_1 + Y - e$	5.3			5.3			5.6					
Effective green time g (s), $g = G + Y - t_L$	74.7			74.7			42.4					
Green ratio , g/C	0.566			0.566			0.321					
Lane group capacity, $c=S*(g/C)$, (veh/h)	1667			1955			1493					
V/c ratio , X	0.493			1.119			0.443					
Flow ratio, V/S	0.279			0.633			0.142					
Critical Lane group/phase (✓)				✓			✓					
Sum of flow ratios for critical lane groups, Y_c $Y_c = \sum$ (critical lane groups, v/s)							0.775					
Total Lost time per cycle, L (s)							10.9					
Critical flow rate to capacity ratio, X_c $X_c = (Y_c)/(C)/(C - L)$							0.845					

Table 5.16: Capacity analysis of the 2nd intersection

In Table 5.17 the computation of the delay and the evaluation of LOS (Level of Service) is reported. In the analysis period there is no initial queue, so $d_3 = 0$. Progression factor (PF) is based on the arrival type.

The first calculation is the delay (and consequently the LOS) of each lane group.

The level of service of the whole intersection is E.




Lane Group Capacity, Control Delay, and LOS Determination				
	EB	WB	NB	SB
Lane Group				
Adjusted flow rate, V (veh/h)	822	2187	662	
V/c ratio, X=V/c	0.493	1.119	0.443	
Total Green ratio, g/C	0.57	0.57	0.32	
Uniform delay, $d_1 = \frac{0.50 C [1 - (g/C)]^2}{1 - [\min(1, X)g/C]}$ (s/veh)	17	28	36	
Incremental Delay Calibration, K	0.5	0.5	0.5	
Incremental delay, d_2 $d_2 = 900T[(X - 1) + \sqrt{(X - 1)^2 + \frac{8KIX}{cT}}]$ (s/veh)	1.0	61.1	1.0	
Initial queue Delay, d_3 (s/veh)	0	0	0	
Uniform Delay, d_1 (s/veh)				
Progression Adjustment Factor, PF	1	1	1	
Delay, $d = d_1(PF) + d_2 + d_3$ (s/veh)	18	89.1	37	
LOS by lane group	B	F	D	
Delay by approach, $d_A = \frac{\sum(d_i)(V_i)}{\sum V_i}$ (s/veh)	18	89.1	37	
LOS by Approach	B	F	D	
Approach flow rate, V_A (veh/h)	822	2187	662	
Intersection delay, $d_I = \frac{\sum(d_A)(V_A)}{\sum V_A}$ (s/veh)	63.8	Intersection LOS		E

Table 5.17: Delay and Level of Service of the 2nd intersection

5.3.3 Junction 3: Via Olevano Romano - via Prenestina - via Prenestina

This Junction is Via Olevano Romano - via Prenestina - via Prenestina which is composed by 3 approaches has a signal setting composed by Two phases.

Figure 5.4 shows a geometrical representation of the junction, with the characteristics reported in Table 5.9.

Approach	Number of lanes	Lane width (m)	Grade %
NB	1	3.5	0
EB	3	3.5	0
WB	3	3.5	0

Table 5.18: Geometrics characteristics of 3rd Intersection

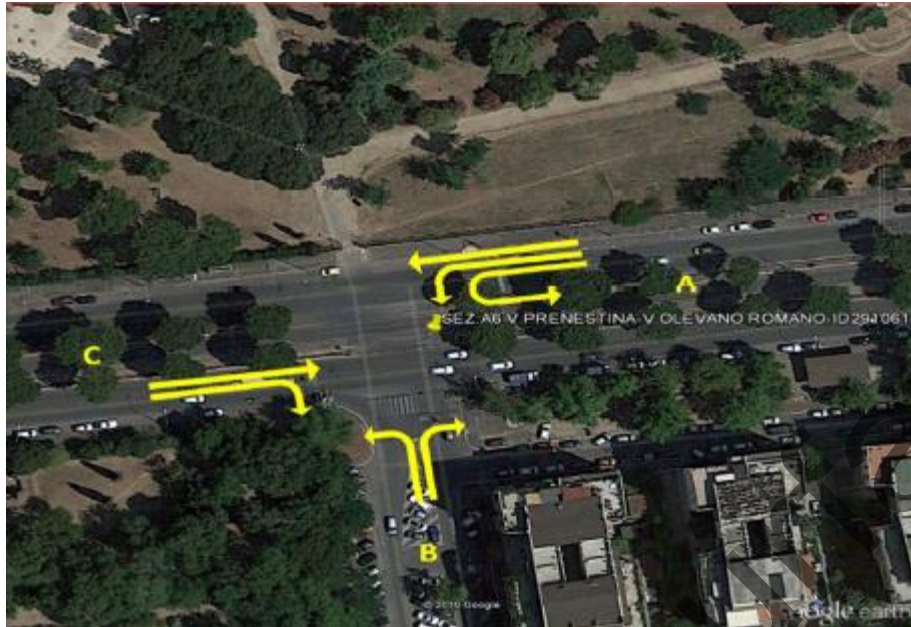


Figure 5.5: Via Olevano Romano - via Prenestina - via Prenestina

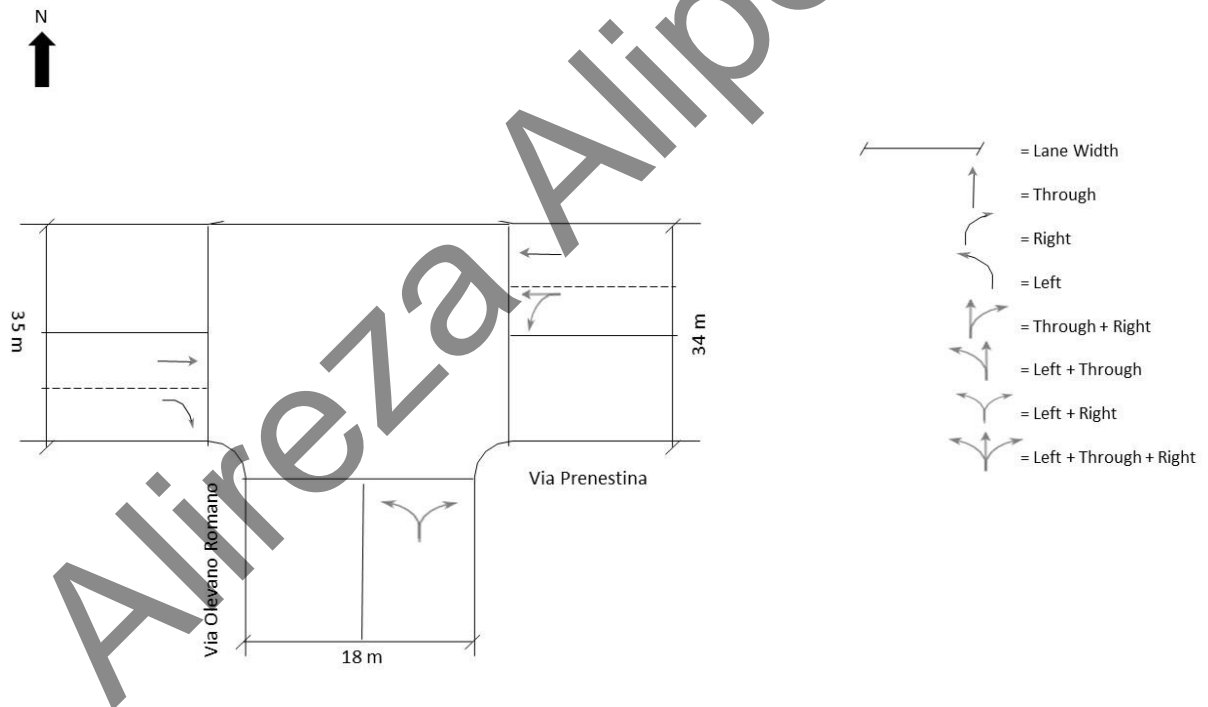


Figure 5.6: Geometric representation of the 3rd intersection

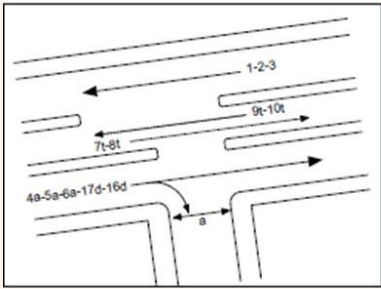
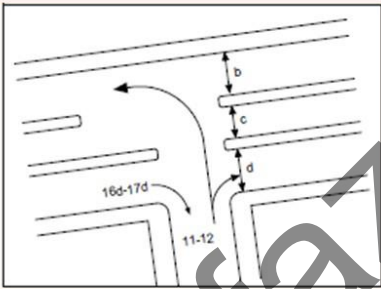
Phases	1	2
Diagram		
Green (s)	75	45
Yellow (s)	4	4
Red (s)	49	79
Cycle (s)	128	

Table 5.19: Phase plane of 2nd intersection

INPUT WORKSHEET																
GENERAL INFORMATION						SITE INFORMATION										
ANALYST	Alireza Alipourfaz					INTERSECTION		Via Olevano Romano - via Prenestina - via Prenestina								
AGENCY OR COMPANY						AREA TYPE		CBD		OTHERS						
DATE PERFORMED	5/7/2020					JURISDICTION										
ANALYSIS TIME PERIOD	8:00 - 8:15					ANALYSIS YEAR		2020								
VOLUME AND TIME INPUT																
	EB			WB			NB			SB						
	LT	TH	RT^1	LT	TH	RT^1	LT	TH	RT^1	LT	TH	RT^1				
Volume, V (veh/h)		1302	263	4	1110		204		60							
Heavy vehicles, % HV	3.3			2.2			1.5									
Peak-hour factor, PHF	0.9			0.9			0.9									
Pretimed (P) or actuated (A)	P			P			P									
Start-up lost time, l1 (s)	2.5			2.5			2.5									
Clearance lost time (s)	2.0			3.0			3.0									
Total lost time (s)	4.5			5.5			5.5									
Arrival type, AT	3			3			3									
Approach pedestrian volume, 2 vped (p/h)	15			15			15									
Approach bicycle volume, 2 vbic (bicycles/h)	N			N			N									
Parking (Y or N)	N			N			N									
Parking maneuvers, Nm (maneuvers/h)	0			0			0									
Bus stopping, NB (buses/h)	N			N			N									
Min. timing for pedestrians, 3 Gp (s)	12.1			12.1			6.25									
SIGNAL PHASING PLAN																
DIAGRAM	ø1		ø2		ø3		ø4		ø5		ø6		ø7		ø8	
Timing	G=	75	G=	45	G=		G=		G=		G=		G=		G=	
	Y=	4	Y=	4	Y=		Y=		Y=		Y=		Y=		Y=	
Permitted Turns					Permitted Turns					Pedestrian		Cycle Length, C= 128 s				

Table 5.20: Volume and timing input of 3rd intersection

In Table 5.20 volume and timing input are summarized. The traffic signal is pretimed. The start-up lost time is assumed to be 2.5 s, which is a feasible value for urban area during the peak hour. Arrival type is assumed 3 corresponding to random arrivals.

Table 5.19 shows the phase planning of the intersection. There is no overlap. The table shows the timing of the signal. The cycle length of this junction is 128 seconds which consist of 2 signal phases. The first phase is for NB approach. The second phase is for the EB/WB approach. The first phases have 75 seconds of green, 4 seconds of yellow, and 49 seconds of red. The second phase has 45 seconds of green, 4 seconds of yellow, and 79 seconds of red. There are 4 seconds of total red which guarantee the clearance of the conflicting area

VOLUME ADJUSTMENT AND SATURATION FLOW RATE WORKSHEET												
General Information												
Project Description: Via Olevano Romano - via Prenestina - via Prenestina												
Volume Adjustment												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Volume, V (veh/h)	0	1302	263	4	1110	0	204	0	60	0	0	0
Peak-hour factor, PHF	0.9			0.9			0.9			0		
Adjusted flow rate, $v_p = V/PHF$ (veh/h)	0	1447	292.22	4	1233.3	0	226.7	0	66.67			
Lane Group	1		1	1			1					
Adjusted flow rate in lane group, v (veh/h)	1447		292.22	1233			293					
Proportion of LT or RT (P_LT, P_RT)		-	1	0.003	-		0.774	-	0.228		-	
Saturation flow rate												
base saturation flow, S_0 (pc/h/ln)	2100			2100			2100					
number of lanes, N	2		1	3			1					
Lane width adjustment factor, f_W	0.989		0.98	0.989			0.990					
Heavy-vehicle adjustment factor, f_{HV}	0.97		0.96	0.978			0.985					
Grade adjustment factor, f_g	1		1	1			1					
Parking adjustment factor, f_p	1		1	1.000			1.000					
Bus blockage adjustment factor, f_{bb}	1		1	1.000			1.000					
Area type adjustment factor, f_a	1		1	1			1					
Lane Utilization adjustment factor, f_{LU}	0.952		1	0.908			1					
Left-Turn adjustment afctor, f_{LT}	1		1	0.92			0.960					
Right-Turn adjustment afctor, f_{RT}	1		0.966	1			0.960					
Left-Turn ped/bike adjustment afctor, f_{Rpb}	0		1	0.71			1					
right-Turn ped/bike adjustment afctor, f_{Rpb}	1		0.976	0			0.948					
Adjusted saturation flow, s (veh/h) $s = s_0 * N * f_w * f_{hv} * f_g * f_{pb} * f_a * f_{LU} * f_{LT} * f_{RT} * f_{lpb} * f_{Rpb}$	3510		1550	3490			1963					

Table 5.21: Volume adjustment and saturation flow rate of the 3rd intersection

In Table 5.21 the capacity analysis of the intersection has been reported. The analysis considered each lane group as a single entity.

The effective green time is the time useful for crossing the conflicting area, so green plus yellow minus the lost time for the phase. It is necessary to determine the capacity of the lane group. Then it's possible to define the volume to capacity ratio, which can indicate, if it's higher than 1, that there is an excess of demand over capacity.

SUPPLEMENTAL WORKSHEET FOR PEDESTRIAN-BICYCLE EFFECTS ON PERMITTED LEFT TURNS AND RIGHT TURNS												
General Information												
Project Description: Via Olevano Romano - via Prenestina - via Prenestina												
Permitted Left Turns												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Effective pedestrian green time, 1, 2 gp (s)						12.08						
Conflicting pedestrian volume, 1 vped (p/h)						15						
vpedg = vped (C/gp)						159						
OCCpedg = vpedg/2000 if (vpedg ≤ 1000) or OCCpedg = 0.4 + vpedg/10,000 if (1000 < vpedg ≤ 5000)						0.079						
Opposing queue clearing green, 3, 4 gq (s)						33.2						
Effective pedestrian green consumed by opposing vehicle queue, gq/gp; if gq ≥ gp then flpb = 1.0						1						
OCCpedu = OCCpedg [1 - 0.5(gq/gp)]						0.020						
Opposing flow rate, 3 vo (veh/h)						27						
OCCr = OCCpedu [e ⁻¹ - (5/3600)vo]						0.098						
Number of cross-street receiving lanes, 1 Nrec						3						
Number of turning lanes, 1 Nturn						3						
ApbT = 1 - OCCr if Nrec = Nturn ApbT = 1 - 0.6(OCCr) if Nrec > Nturn						0.902						
Proportion of left turns, 5 PLT						0.774						
Proportion of left turns using protected phase, 6 PLTA						0						
flpb = 1.0 - PLT(1 - ApbT)(1 - PLTA)						0.705						
Permitted Right Turns												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Effective pedestrian green time, 1, 2 gp (s)						12.08			6.25			
Conflicting pedestrian volume, 1 vped (p/h)						20			20			
Conflicting bicycle volume, 1, 7 vbic (bicycles/h)						5			5			
vpedg = vped (C/gp)						1.66			409			
OCCpedg = vpedg/2000 if (vpedg ≤ 1000), or OCCpedg = 0.4 + vpedg/10,000 if (1000 < vpedg ≤ 5000)						0.002			0.205			
Effective green, 1 g (s)						73.5			43.5			
vbicg = vbic (C/g)						8.707			14.71			
OCCbicg = 0.02 + vbicg/2700						0.023			0.020			
OCCr = OCCpedg + OCCbicg - (OCCpedg)(OCCbicg)						0.023			0.224			
Number of cross-street receiving lanes, 1 Nrec						3			3			
Number of turning lanes, 1 Nturn						3			3			
ApbT = 1 - OCCr if Nrec = Nturn ApbT = 1 - 0.6(OCCr) if Nrec > Nturn						0.97			0.770			
Proportion of right turns, 5 PRT						1			0.22			
Proportion of right turns using protected phase, 8 PRTA						0			0			
fRpb = 1.0 - PRT(1 - ApbT)(1 - PRTA)						0.97			0.948			

 Table 5.22: Supplemental Table for Pedestrian-Bicycle effects on Permitted Right Turns of the 3rd intersection

SUPPLEMENTAL WORKSHEET FOR PERMITTED LEFT TURNS OPPOSED BY MULTILANE APPROACH												
General Information												
Project Description: Via Olevano Romano - via Prenestina - via Prenestina												
Input												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Cycle length, C (s)					128							
Total actual green time for LT lane group, 1 G (s)					75							
Effective permitted green time for LT lane group, 1 g (s)					74.5							
Opposing effective green time, go (s)					74.5							
Number of lanes in LT lane group, 2 N					3							
Number of lanes in opposing approach, No					2							
Adjusted LT flow rate, vLT (veh/h)					4							
Proportion of LT volume in LT lane group, 3 PLT					0.004							
Adjusted flow rate for opposing approach, vo (veh/h)					1447							
Lost time for LT lane group, tL					5.5							
Computation												
	EB			WB			NB			SB		
LT volume per cycle, LTC = vLT.C/3600					0.158							
Opposing lane utilization factor, fLUo (refer to Volume Adjustment and Saturation Flow Rate Worksheet)					0.908							
Opposing flow per lane, per cycle $v_{olc} = \frac{v_o C}{3600 N_o f_{LUo}} \quad (\text{veh/C/ln})$					28.34							
$g_l = G[e^{-0.882(LTC^{0.717})}] - t_L \quad g_l \leq g \quad (\text{except for exclusive left-turn lanes})^{1, 4}$					53.79							
Opposing platoon ratio, Rpo (refer to Exhibit 16-11)					1							
Opposing queue ratio, qro = max[1 - Rpo(go/C), 0]					0.414							
$q_q = \frac{v_{olc} q_{ro}}{0.5 - [v_{olc}(1 - q_{ro})/g_o]} - t_L \quad v_{olc}(1 - q_{ro})/g_o \leq 0.49$ (note case-specific parameters) ¹					36.579							
$g_u = g - g_q \text{ if } g_q \geq g_f, \text{ or}$ $g_u = g - g_f \text{ if } g_q < g_f$					21.204							
EL1 (refer to Exhibit C16-3)					4.7							
$P_L = P_{LT} \left[1 + \frac{(N-1)g}{(g_l + g_u/E_{L1} + 4.24)} \right]$ (except with multilane subject approach) ⁵					0.065							
fmin = 2(1 + PL)/g					0.028							
$f_m = [g_f/g] + [g_u/g] \left[\frac{1}{1 + P_L(E_{L1} - 1)} \right], (f_{min} \leq f_m \leq 1.00)$					0.945							
fLT = [fm + 0.91(N - 1)]/N (except for permitted left turns) ⁶					0.922							

Table 5.23: SUPPLEMENTAL WORKSHEET FOR PERMITTED LEFT TURNS OPPOSED BY MULTILANE APPROACH of the 3rd intersection

CAPACITY AND LOS WORKSHEET												
General Information												
Project Description: Via Olevano Romano - via Prenestina - via Prenestina												
Capacity Analysis												
	EB			WB			NB			SB		
	LT	TH	RT	LT	TH	RT	LT	TH	RT	LT	TH	RT
Phase number	1		1	1			2					
Phase type	P			P			P					
Lane Group	1		1	1			1					
Adjusted flow rate, V (veh/h)	1447		292	1237			293					
Saturation flow rate, S (veh/h)	3510		1550	3489			1963					
Lost time t_L (s), $t_L = I_1 + Y - e$	5.5		5.5	5.5			5.5					
Effective green time g (s), $g = G + Y - t_L$	73.5		73.5	73.5			43.5					
Green ratio, g/C	0.57		0.57	0.570			0.330					
Lane group capacity, $c = S \cdot (g/C)$, (veh/h)	2015		980	2003			667					
V/c ratio, X	0.72		0.33	0.620			0.440					
Flow ratio, V/S	0.41		0.19	0.350			0.149					
Critical Lane group/phase (\checkmark)	\checkmark						\checkmark					
Sum of flow ratios for critical lane groups, Y_c $Y_c = \sum (\text{critical lane groups, } v/s)$	0.560											
Total Lost time per cycle, L (s)	11											
Critical flow rate to capacity ratio, X_c $X_c = (Y_c)/(C)/(C - L)$	0.614											

Table 5.24: Capacity analysis of the 3rd intersection

In Table 5.25 the computation of the delay and the evaluation of LOS (Level of Service) is reported. In the analysis period there is no initial queue, so $d_3 = 0$. Progression factor (PF) is based on the arrival type.

The first calculation is the delay (and consequently the LOS) of each lane group.

The level of service of the whole intersection is C.

Lane Group Capacity, Control Delay, and LOS Determination						
	EB		WB	NB	SB	
Lane Group	1	1	1	1		
Adjusted flow rate, V (veh/h)	1447	292	1237	293		
V/c ratio, X=V/c	0.72	0.33	0.620	0.440		
Total Green ratio, g/C	0.57	0.57	0.57	0.33		
Uniform delay, $d_1 = \frac{0.50 C [1 - (g/C)]^2}{1 - [\min(1, X)g/C]}$ (s/veh)	19.74	14.3	18	35		
Incremental Delay Calibration, K	0.5	0.5	0.5	0.5		
Incremental delay, d_2 $d_2 = 900T[(X-1) + \sqrt{(X-1)^2 + \frac{8kIX}{cT}}]$ (s/veh)	2.23	0.99	1.5	2.1		
Initial queue Delay, d_3 (s/veh)	0	0	0	0		
Uniform Delay, d_1 (s/veh)						
Progression Adjustment Factor, PF	1	1	1	1		
Delay, $d = d_1(PF) + d_2 + d_3$ (s/veh)	21.97	15.3	19.5	37.1		
LOS by lane group	C	B	B	D		
Delay by approach, $d_A = \frac{\sum(d_i)(V_i)}{\sum V_i}$ (s/veh)	20.85		19.5	37.1		
LOS by Approach	C		B	D		
Approach flow rate, V_A (veh/h)	1447		1237	293		
Intersection delay, $d_I = \frac{\sum(d_A)(V_A)}{\sum V_A}$ (s/veh)	21.9		Intersection LOS		C	

Table 5.17: Delay and Level of Service of the 3rd intersection

6. Optimization Methods

Optimization methods are designed to provide the 'best' values of system design and operating policy variables. They could be either minimizing or maximizing. There are many cases of study, depending on the type of the Objective Function: in this project, there is a Non-Linear Problem (NLP), so a Linear Optimization cannot be utilized.

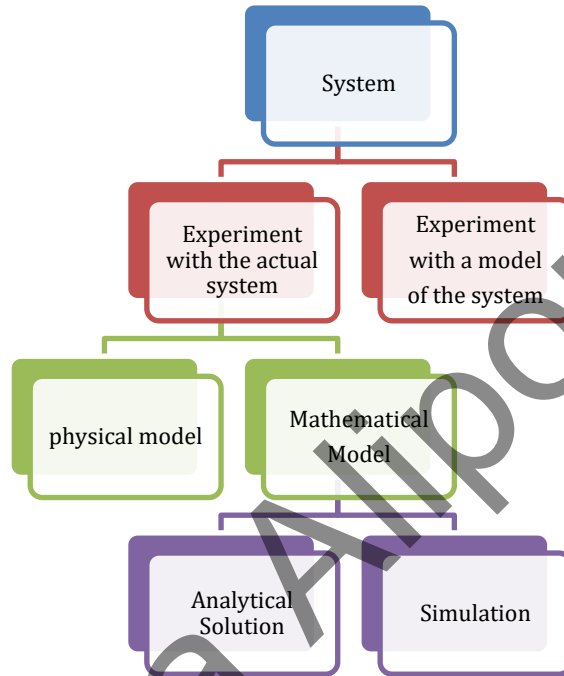


Figure 6.1: Diagram of system optimization

6.1. Minimum Cycle

The minimum cycle length is the one which complies to the capacity constraints taken as equality:

$$g_i = y_i \cdot C$$

$$C_{\min} = \frac{\sum_i l_i}{1 - \sum_i y_i}$$

Where:

l_i = lost time for the phase i ;

y_i = critical flow ratio for phase i ;

The capacity constraints ensure that the vehicles arrived are served.

6.2. Optimum Cycle

Webster (1958) proposed an equation for the calculation of cycle length seeking to minimize vehicle delay. This results in the optimum for the function of Webster, which is different from the HCM one, and is obtained under the assumptions of:

- arrivals according to Poisson process
- intersection not oversaturated
- flows not very unbalanced

The optimum cycle length formula is as follows:

$$C_{opt} = \frac{(1.5 \times \sum_i l_i) + 5}{1 - \sum_i y_i}$$

By calculating the optimum cycle length, the minimum effective green can be obtained by this formula.

$$g_i = \frac{y_i}{\sum_i y_i} \cdot (C_{opt} - \sum_i y_i)$$

6.3. Brute-Force Search

In computer science, brute-force search or exhaustive search, also known as generate and test, is a very general problem-solving technique and algorithmic paradigm that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement.

For example a brute-force algorithm to find the divisors of a natural number n would enumerate all integers from 1 to n , and check whether each of them divides n without remainder. A brute-force approach for the eight queens puzzle would examine all possible arrangements of 8 pieces on the 64-square chessboard, and, for each arrangement, check whether each (queen) piece can attack any other.

While a brute-force search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions – which in many practical problems tends to grow very quickly as the size of the problem increases. Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed.

This is the case, for example, in critical applications where any errors in the algorithm would have very serious consequences; or when using a computer to prove a mathematical theorem. Brute-force search is also useful as a baseline method when benchmarking other algorithms or metaheuristics. Indeed, brute-force search can be viewed as the simplest metaheuristic. Brute force search should not be confused with backtracking, where large sets of solutions can be discarded without being explicitly enumerated. The brute-force method for finding an item in a table – namely, check all entries of the latter, sequentially – is called linear search.

6.3.1. Basic Algorithm

In order to apply brute-force search to a specific class of problems, one must implement four procedures, first, next, valid, and output. These procedures should take as a parameter the data P for the particular instance of the problem that is to be solved, and should do the following:

1. first (P): generate a first candidate solution for P .
2. next (P, c): generate the next candidate for P after the current one c .
3. valid (P, c): check whether candidate c is a solution for P .
4. output (P, c): use the solution c of P as appropriate to the application.

6.4. Heuristic

In computer science, artificial intelligence, and mathematical optimization, a heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution. This is achieved by trading optimality, completeness, accuracy, or precision for speed. In a way, it can be considered a shortcut.

A heuristic function, also called simply a heuristic, is a function that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow.

6.4.1. Definition and Motivation

The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand. This solution may not be the best of all the solutions to this problem, or it may simply approximate the exact solution. But it is still valuable because finding it does not require a prohibitively long time.

Heuristics may produce results by themselves, or they may be used in conjunction with optimization algorithms to improve their efficiency.

Results about NP-hardness in theoretical computer science make heuristics the only viable option for a variety of complex optimization problems that need to be routinely solved in real-world applications.

Heuristics underlie the whole field of Artificial Intelligence and the computer simulation of thinking, as they may be used in situations where there are no known algorithms.

6.4.2. Genetic Algorithm

In computer science and operations research, a genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection. John Holland introduced genetic algorithms in 1960 based on the concept of Darwin's theory of evolution, and his student David E. Goldberg further extended GA in 1989. Main part of the code is attached in appendix A.

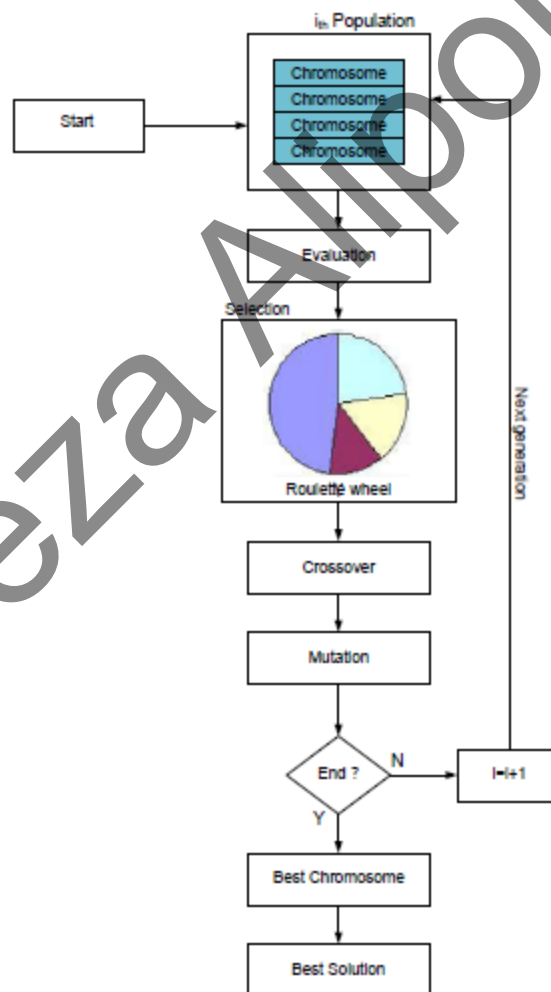


Figure 6.2: Genetic Algorithm Flow Chart

6.4.2.1. Methodology – Optimization Problems

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0 and 1, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form

representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

6.4.2.2. Initialization

The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

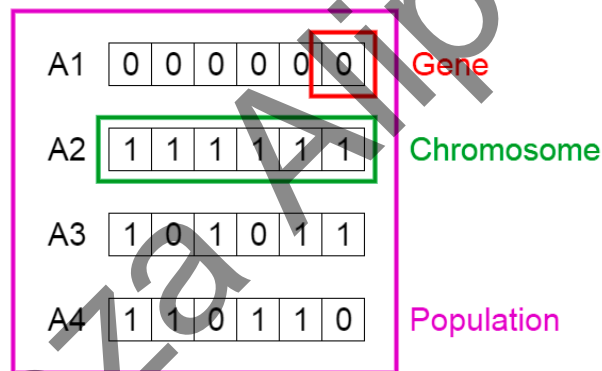


Figure 6.3: Population, Chromosomes and Genes

In this case we have a n -cell chromosome that based on the number of phases (corresponding to green time of each phase) + 1 (corresponding to cycle length).

Cycle length	G1	G2	...	Gn
--------------	----	----	-----	----

Figure 6.4: Chromosomes and Genes

6.4.2.3. Selection

During each successive generation, a portion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent.

6.4.2.4. Roulette Wheel Selection

Fitness proportionate selection, also known as roulette wheel selection, is a genetic operator used in genetic algorithms for selecting potentially useful solutions for recombination.

In fitness proportionate selection, as in all selection methods, the fitness function assigns a fitness to possible solutions or chromosomes. This fitness level is used to associate a probability of selection with each individual chromosome. If f_i is the fitness of individual i in the population, its probability of being selected is

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j},$$

where N is the number of individuals in the population.

This could be imagined similar to a Roulette wheel in a casino. Usually a proportion of the wheel is assigned to each of the possible selections based on their fitness value. This could be achieved by dividing the fitness of a selection by the total fitness of all the

selections, thereby normalizing them to 1. Then a random selection is made similar to how the roulette wheel is rotated.

While candidate solutions with a higher fitness will be less likely to be eliminated, there is still a chance that they may be eliminated because their probability of selection is less than 1 (or 100%). Contrast this with a less sophisticated selection algorithm, such as truncation selection, which will eliminate a fixed percentage of the weakest candidates. With fitness proportionate selection there is a chance some weaker solutions may survive the selection process. This is because even though the probability that the weaker solutions will survive is low, it is not zero which means it is still possible they will survive; this is an advantage, because there is a chance that even weak solutions may have some features or characteristics which could prove useful following the recombination process.

The **fitness function** determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a **fitness score** to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

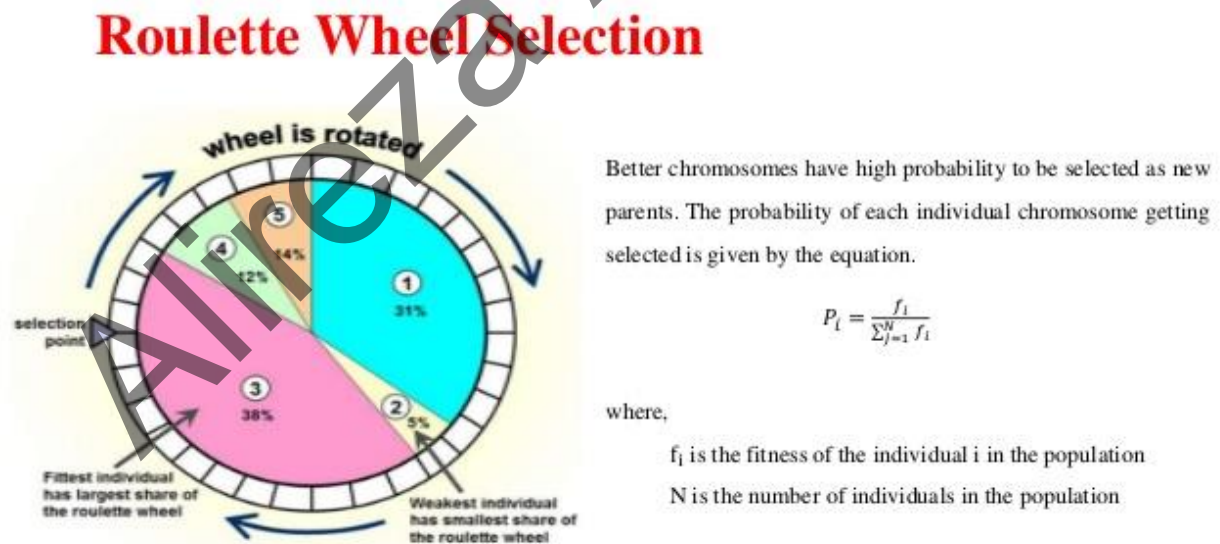


Figure 6.5: Roulette Wheel Selection

6.4.2.5. Genetic Operators

The next step is to generate a second generation population of solutions from those selected through a combination of genetic operators: crossover (also called recombination), and mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests that more than two "parents" generate higher quality chromosomes.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally, the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions. These fewer fit solutions ensure genetic diversity within the genetic pool of the parents and therefore ensure the genetic diversity of the subsequent generation of children.

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a **crossover point** is chosen at random from within the genes.

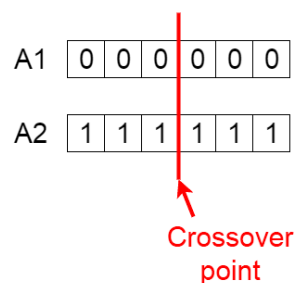


Figure 6.6: Crossover Point

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

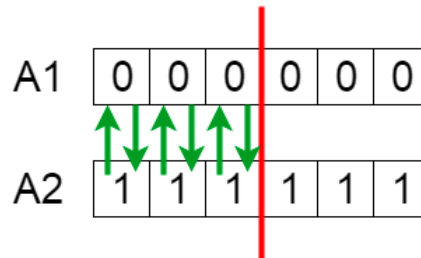
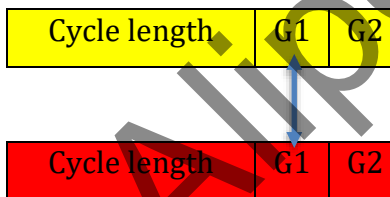


Figure 6.7: Exchanging Genes among Parents

Case 1

Parents



Then we calculate G2 based on the constraints.

Children

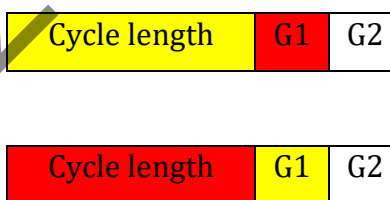


Figure 6.8: Exchanging Genes among Parents

Case 2

Parents



Then we calculate G1 based on the constraints.

Children

Cycle length	G1	G2
--------------	----	----

Cycle length	G1	G2
--------------	----	----

Figure 6.9: Exchanging Genes among Parents

Case 3

Parents

Cycle length	G1	G2
--------------	----	----

Cycle length	G1	G2
--------------	----	----

Then we calculate G2 based on the constraints.

Children

Cycle length	G1	G2
--------------	----	----

Cycle length	G1	G2
--------------	----	----

Figure 6.10: Exchanging Genes among Parents

In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability. This implies that some of the bits in the bit string can be flipped.

Before Mutation

A5	1	1	1	0	0	0
----	---	---	---	---	---	---

After Mutation

A5	1	1	0	1	1	0
----	---	---	---	---	---	---

Figure 6.11: Mutation: Before and After

6.4.2.6. Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached (We used this method in this project.)
- Allocated budget (computation time/money) reached
- The highest-ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

Stopping criteria is defined based on maximum iteration which for all intersection solution has been found by 100 iterations. As an advantage of this method rather than previous one (Brute Force algorithm), although the smallest steps are defined for cycle length and green ratio, shorter running time is required to find optimum solution. Main part of the codes is attached in appendix.

6.4.3. Hill Climbing Algorithm

Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence. Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem. This solution may not be the global optimal maximum. But in this case, we only have one minimum which is the global minimum.

In numerical analysis, hill climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by making an incremental change to the solution. If the change produces a better solution, another incremental change is made to the new solution, and so on until no further improvements can be found.

Hill climbing finds optimal solutions for convex problems – for other problems it will find only local optima (solutions that cannot be improved upon by any neighboring configurations), which are not necessarily the best possible solution (the global optimum) out of all possible solutions (the search space).

The relative simplicity of the algorithm makes it a popular first choice amongst optimizing algorithms. It is used widely in artificial intelligence, for reaching a goal state from a starting node. Different choices for next nodes and starting nodes are used in related algorithms. Hill climbing can often produce a better result than other algorithms when the amount of time available to perform a search is **limited**, such as with real-time systems, so long as a small number of increments typically converges on a good solution (the optimal solution or a close approximation). Main part of the code is attached in appendix B.

Simple Hill climbing: It examines the neighboring nodes one by one and selects the first neighboring node which optimizes the current cost as next node.

Algorithm for Simple Hill climbing:

Step 1: Evaluate the initial state. If it is a goal state then stop and return success. Otherwise, make initial state as current state.

Step 2: Loop until the solution state is found or there are no new operators present which can be applied to the current state.

a) Select a state that has not been yet applied to the current state and apply it to produce a new state.

b) Perform these to evaluate new state

- i. If the current state is a goal state, then stop and return success.
- ii. If it is better than the current state, then make it current state and proceed further.
- iii. If it is not better than the current state, then continue in the loop until a solution is found.

Step 3: Exit.

In this case the neighbors are like the figure below:

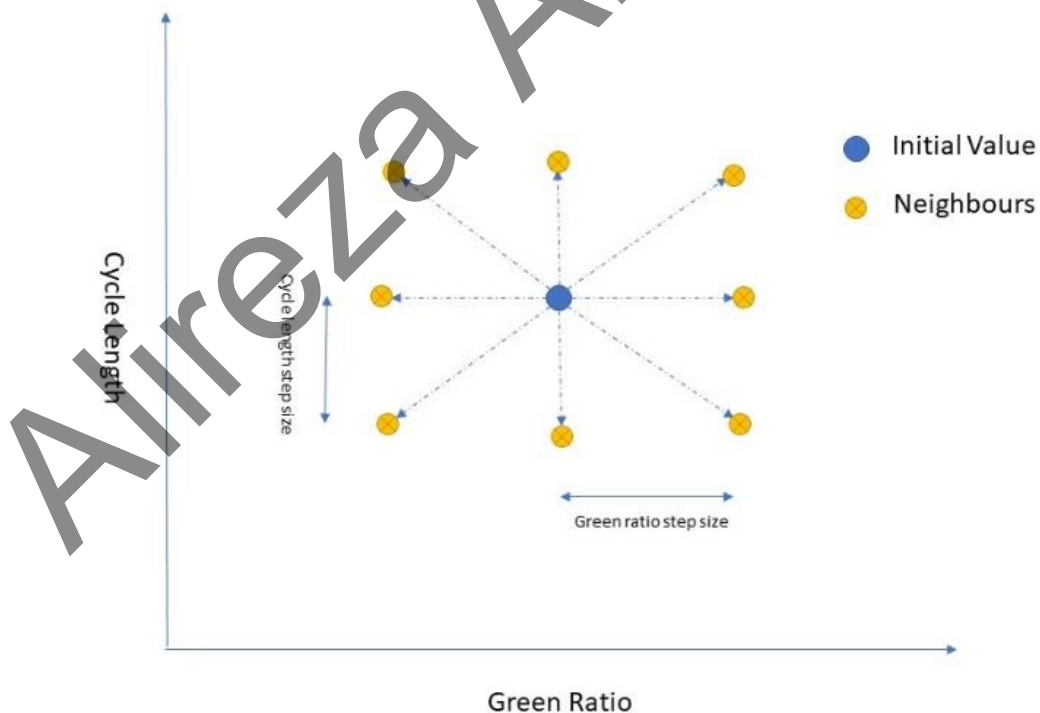


Figure 6.12: Generation Neighbors

7. Optimization

The aim of this chapter is to minimize the delay of each single intersection, considered without interaction with the close ones.

7.2. Optimization of the 1st Intersection

The first intersection is Viale Ronchi - Largo Irpinia - via Prenestina - via Prenestina. The current overall delay of the 1st intersection is calculated about 41.8 [sec] and the level of service of intersection is D.

The first step is to consider the capacity constraints to obtain the minimum cycle length to serve all the demand and its respective green duration; this is reported in Table together with the optimum cycle for Webster. The delay for the Minimum Cycle is 45.77 (sec), and for Webster Optimum Cycle is 35.34 (sec).

By implementing the minimization of the delay problem in the MATLAB, we have the opportunity of investigating cycle length and green ratio variation simultaneously. but it should be in the boundaries.

By having total lost time and the flow of lane grouping from the first part of the project we can calculate C_{min} . This is the lowest possible value for the minimum cycle and upper bound for cycle variation we used a value for example 5 experimentally because you have explained that in the real world we don't have a cycle like for example 10 min. so by defining a loop over the cycle length for each step we can have an internal loop that is responsible for green ratio variation.

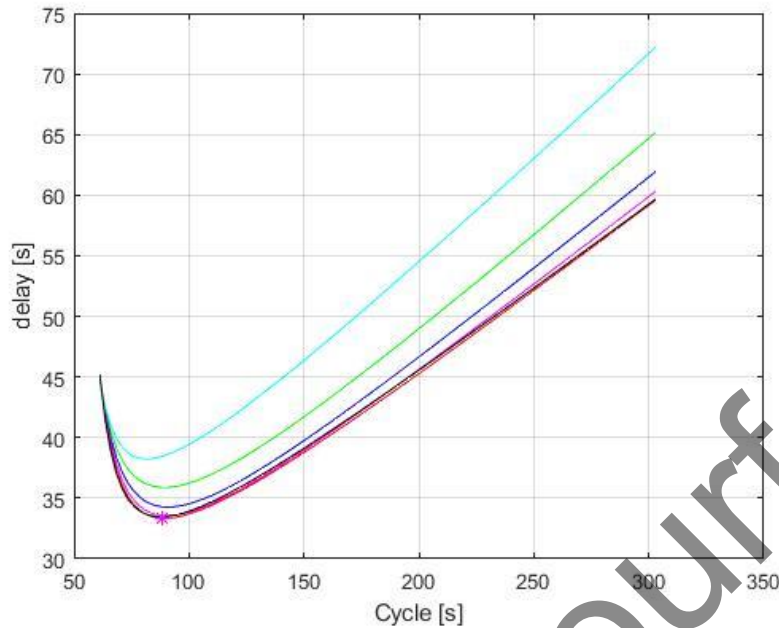


Figure 7.1: Variation of delay with respect to C keeping G1/G2 unvaried

In Figure The best solution, corresponding to a delay of 33.18 seconds, refers to a cycle length of 91.31 seconds. Each color represents a different green ratio in the figure.

Green ratio

for determining the lower and upper boundary of the green ratio the variation we need to derive the formula that links cycle length and green split. Based on this formula:

$\frac{g1}{g2} = \frac{y1}{y2}$ and if we have maximum for $y2$, we will have a minimum value for the green

split. So, by $C_{min} = \frac{L}{1-y}$, we can derive $y2$ which means the critical value of the second phase. In the same way for having a maximum green split, we derive $y1$ based on the minimum cycle to have the maximum number in numerator.

So:

Minimum value for $(g1/g2) = y1/(1 - y1 - L/C)$

Minimum value for $(g1/g2) = (1 - y2 - L/C)/y2$

in this way, we can calculate the minimum delay by considering cycle length and green ratio simultaneously.

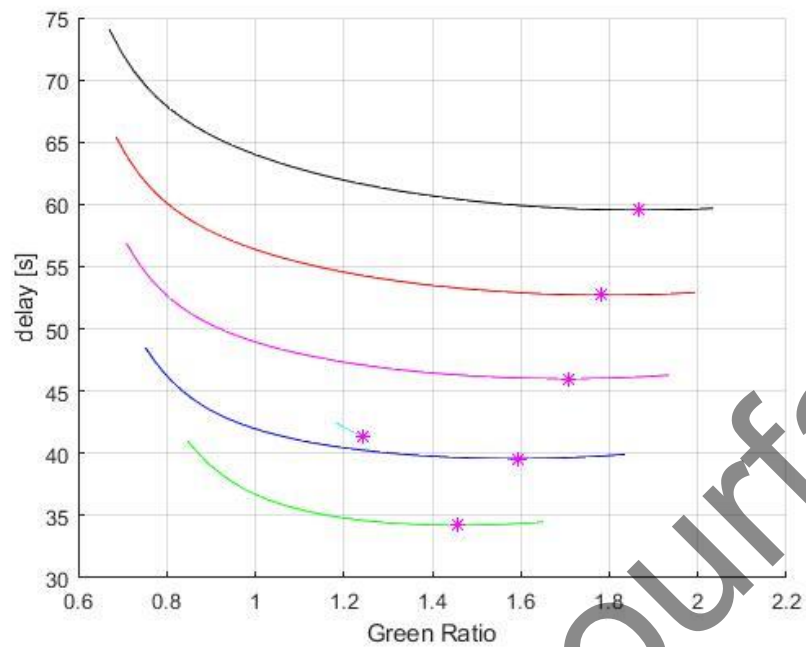


Figure 7.2: Variation of delay with variation of Green ratio

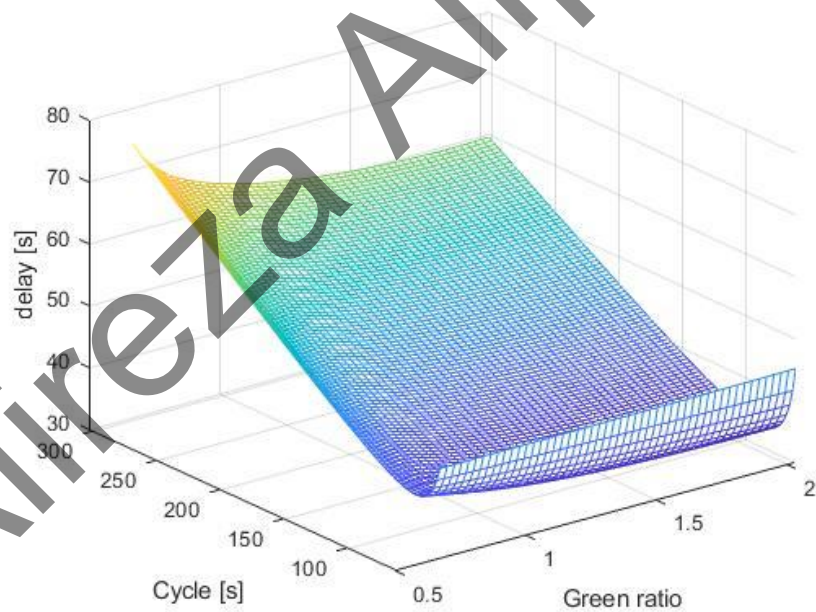


Figure 7.3: Variation of delay with variation of Green ratio and Cycle time

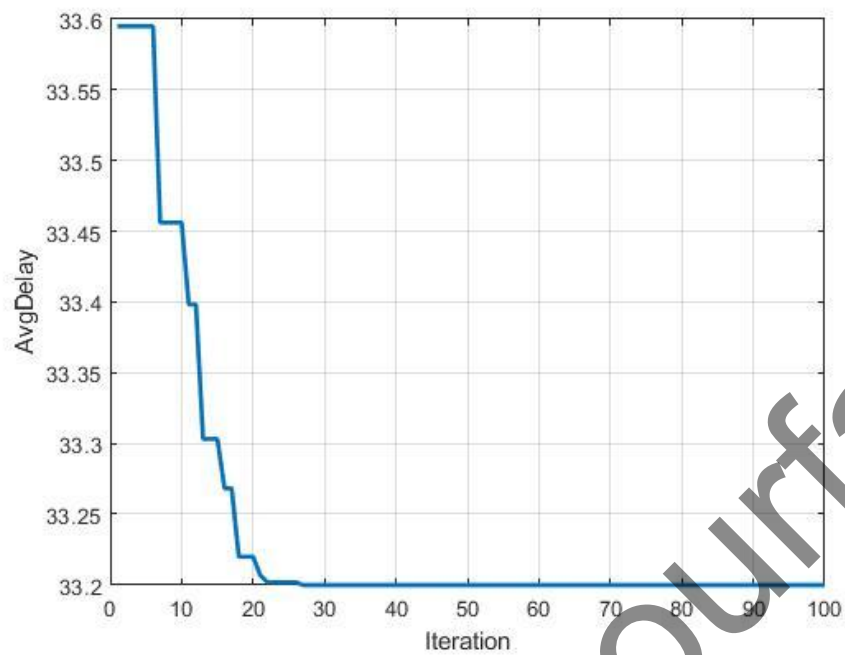


Figure 7.4: Evolution of the minimum Delay for every generation (Genetic Algorithm)

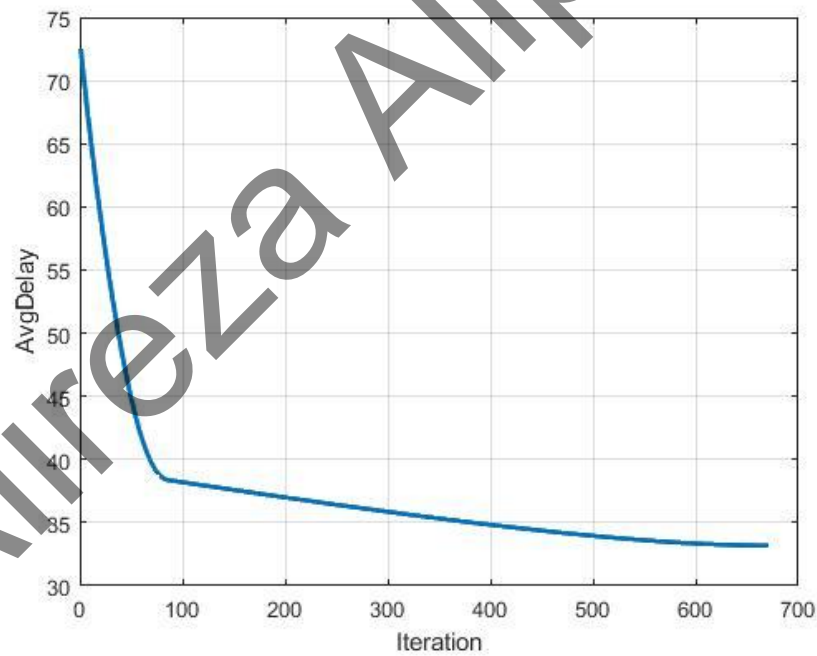


Figure 7.5: Evolution of the minimum Delay for every generation (Hill Climbing Algorithm)

The 2 figures in previous slide presents optimized delay at every iteration corresponding green splits and cycle length for the intersection. Green splits and cycle length were the input decision variables for both algorithms and were programmed and controlled to have values in the boundaries. The objective was to select the best possible combination of green splits and cycle length in response approaching traffic volume to minimize the objective function i.e., average vehicle delay at the intersection.

	Current Situation	Minimum Cycle	Webster optimum Cycle	Brute Force	Genetic Algorithm	Hill Climbing Algorithm
Cycle	132	60.61	110.23	91.31	93.24	91.26
Intersection Delay	41.8	45.77	35.34	33.18	33.19	33.18
LOS	D	D	D	C	C	C
g1	42.2	17.88	37.63	33.03	33.9	32.32
g2	32.2	14.67	30.87	23.19	23.69	22.88
g3	35.9	12.36	26.02	19.38	19.95	19.34

Table 7.1: Comparison of different methods - 1st intersection

	Current Situation	Minimum Cycle	Webster optimum Cycle	Brute Force	Genetic Algorithm	Hill Climbing Algorithm
Run time (sec)	-	-	-	350	2.18	0.53
Processor	Intel® Core™ i7-5500 U CPU @ 2.4GHz (4CPUs)					
Memory	8 GB					
Graphic Card	AMD Radeon R7 M265 Series					

Table 7.2: Comparison of different methods - 1st intersection

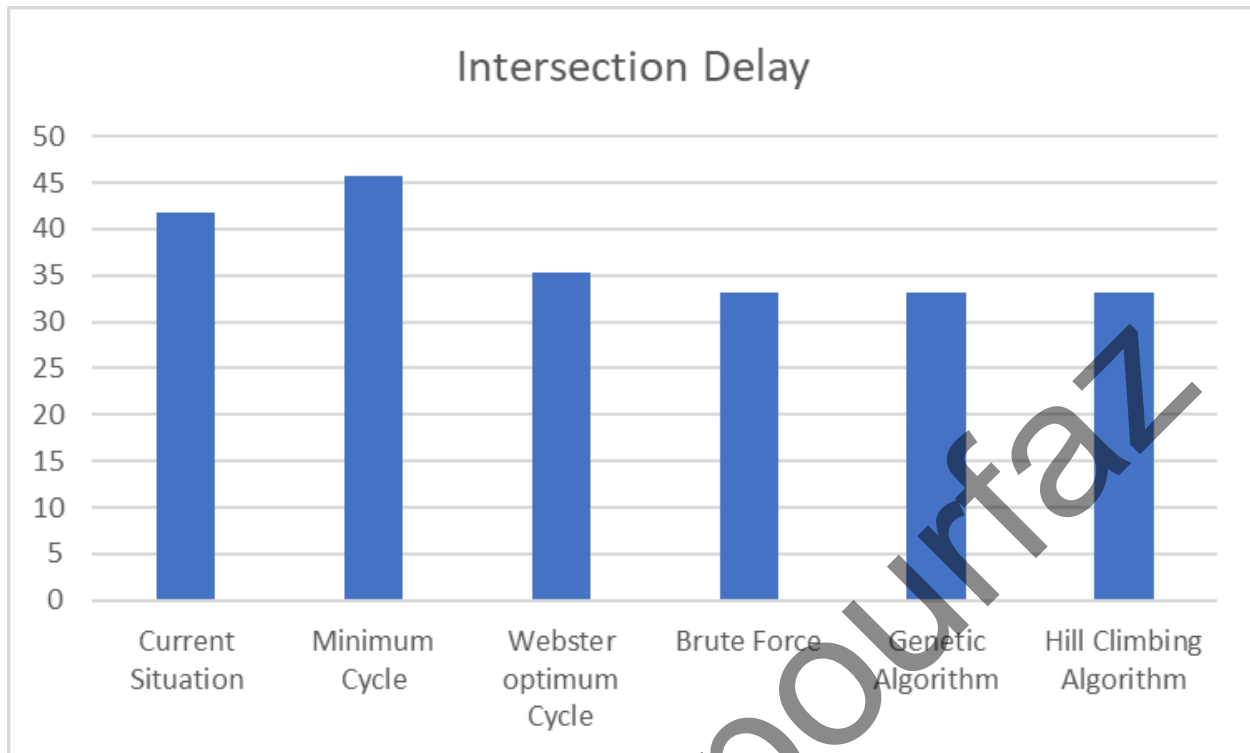


Table 7.2.1: Comparison Between Different Methods

7.3. Optimization of the 2nd Intersection

The second intersection is Via Dignano d'Istria - Largo Irpinia - via Prenestina – via Prenestina. The current overall delay of the 2nd intersection is calculated about 63.8 [sec] and the level of service of intersection is E.

The first step is to consider the capacity constraints to obtain the minimum cycle length to serve all the demand and its respective green duration; this is reported in Table together with the optimum cycle for Webster. The delay for the Minimum Cycle is 28.14 (sec), and for Webster Optimum Cycle is 25.06 (sec).

By implementing the minimization of the delay problem in the MATLAB, we have the opportunity of investigating cycle length and green ratio variation simultaneously. but it should be in the boundaries.

By having total lost time and the flow of lane grouping from the first part of the project we can calculate Cmin. This is the lowest possible value for the minimum cycle and upper bound for cycle variation we used a value for example 5 experimentally because you have explained that in the real world we don't have a cycle like for example 10 min. so by defining a loop over the cycle length for each step we can have an internal loop that is responsible for green ratio variation.

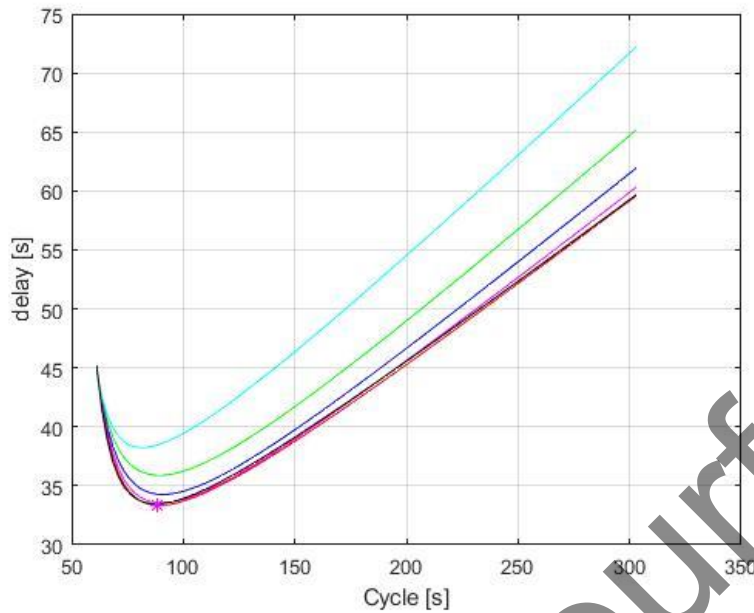


Figure 7.6: Variation of delay with respect to C keeping G1/G2 unvaried

In Figure The best solution, corresponding to a delay of 18.97 seconds, refers to a cycle length of 80.84 seconds. Each color represents a different green ratio in the figure.

Green ratio

for determining the lower and upper boundary of the green ratio the variation we need to derive the formula that links cycle length and green split. Based on this formula:

$\frac{g1}{g2} = \frac{y1}{y2}$ and if we have maximum for $y2$, we will have a minimum value for the green

split. So, by $C_{min} = \frac{L}{1-y}$, we can derive $y2$ which means the critical value of the second phase. In the same way for having a maximum green split, we derive $y1$ based on the minimum cycle to have the maximum number in numerator.

So:

Minimum value for $(g1/g2) = y1/(1 - y1 - L/C)$

Minimum value for $(g1/g2) = (1 - y2 - L/C)/y2$

in this way, we can calculate the minimum delay by considering cycle length and green ratio simultaneously.

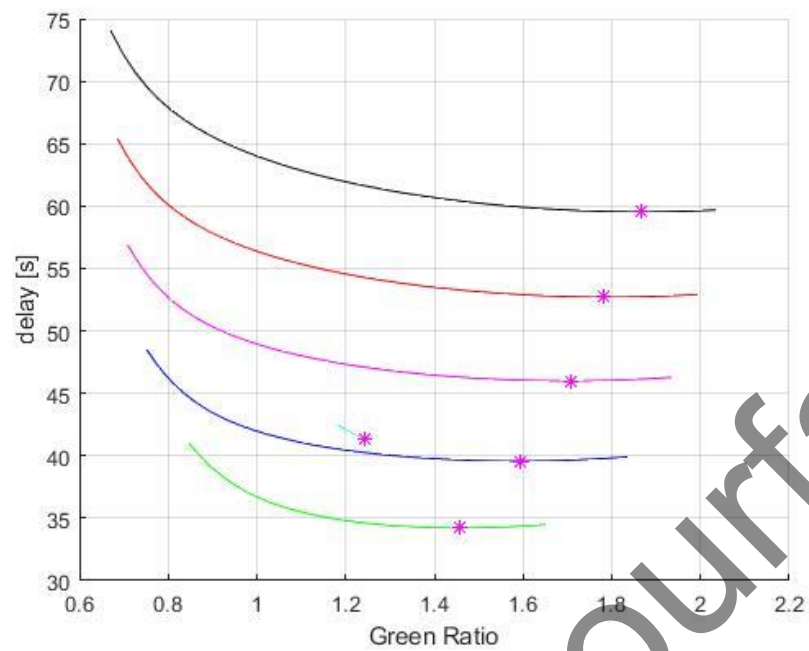


Figure 7.7: Variation of delay with variation of Green ratio

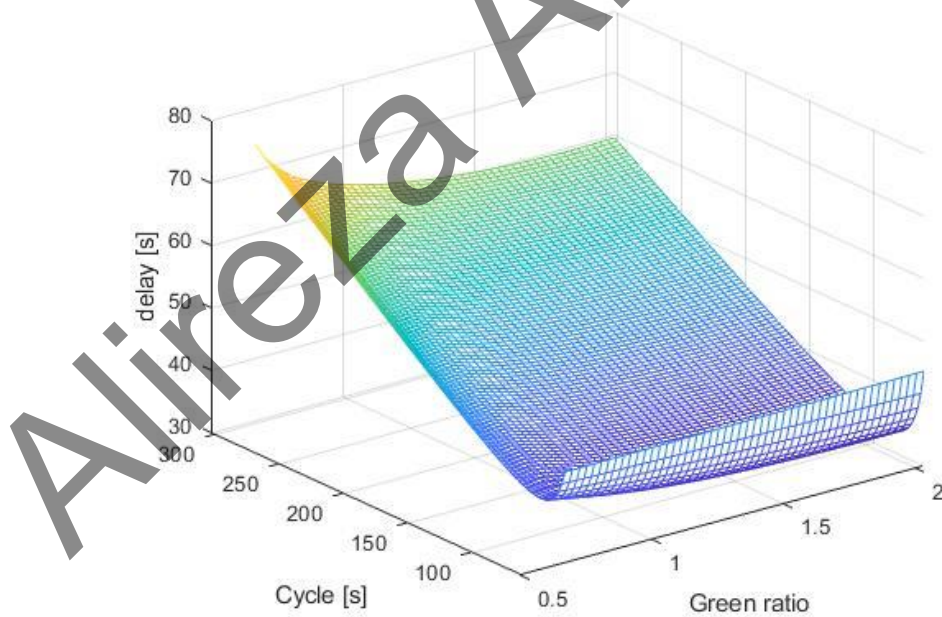


Figure 7.8: Variation of delay with variation of Green ratio and Cycle time

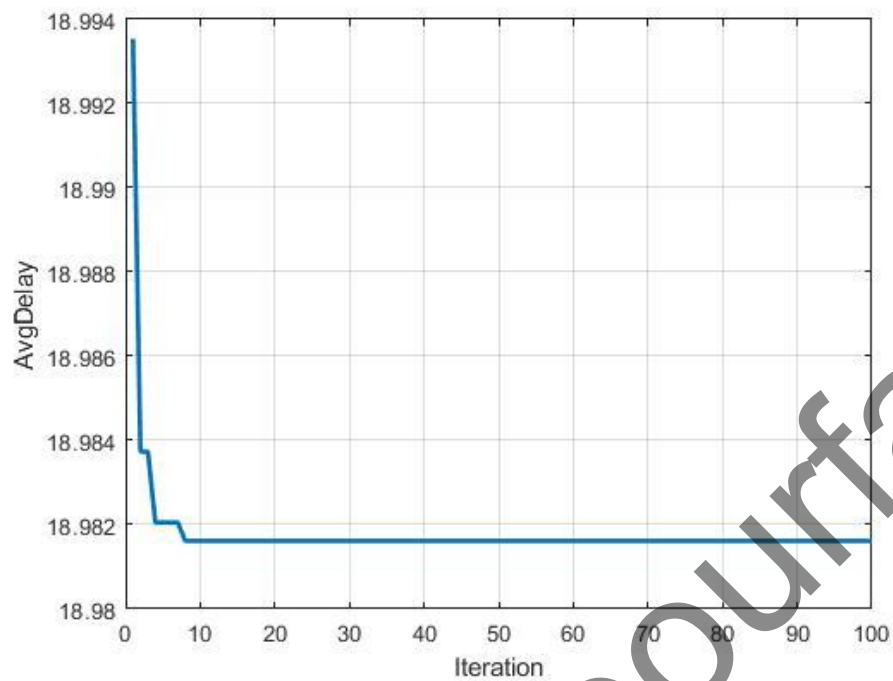


Figure 7.9: Evolution of the minimum Delay for every generation (Genetic Algorithm)

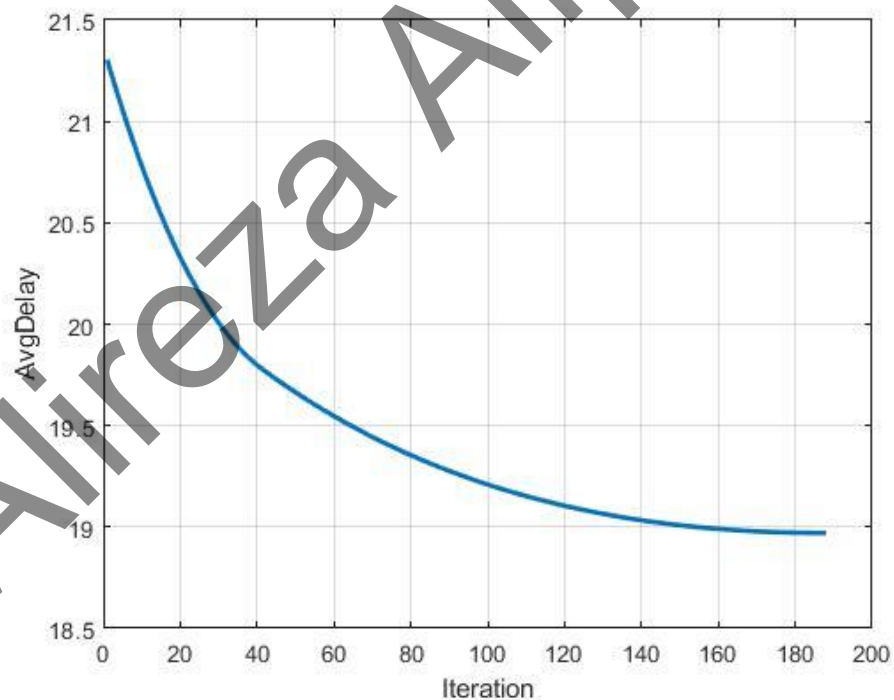


Figure 7.10: Evolution of the minimum Delay for every generation (Hill Climbing Algorithm)

The 2 figures in previous slide presents optimized delay at every iteration corresponding green splits and cycle length for the intersection. Green splits and cycle length were the input decision variables for both algorithms and were programmed and controlled to have values in the boundaries. The objective was to select the best possible combination of green splits and cycle length in response approaching traffic volume to minimize the objective function i.e., average vehicle delay at the intersection.

	Current Situation	Minimum Cycle	Webster optimum Cycle	Brute Force	Genetic Algorithm	Hill Climbing Algorithm
Cycle	132	48.44	94.88	80.84	80.71	80.63
Intersection Delay	63.8	28.14	25.06	18.97	18.98	18.97
LOS	E	C	C	B	B	B
g1	74.7	30.66	64.67	56.57	56.18	56.43
g2	42.4	6.88	14.51	13.36	13.63	13.3

Table 7.3: Comparison of different methods - 2nd intersection

	Current Situation	Minimum Cycle	Webster optimum Cycle	Brute Force	Genetic Algorithm	Hill Climbing Algorithm
Run time (sec)	-	-	-	115	3.32	1.53
Processor	Intel® Core™ i7-5500 U CPU @ 2.4GHz (4CPUs)					
Memory	8 GB					
Graphic Card	AMD Radeon R7 M265 Series					

Table 7.4: Comparison of different methods - 2nd intersection

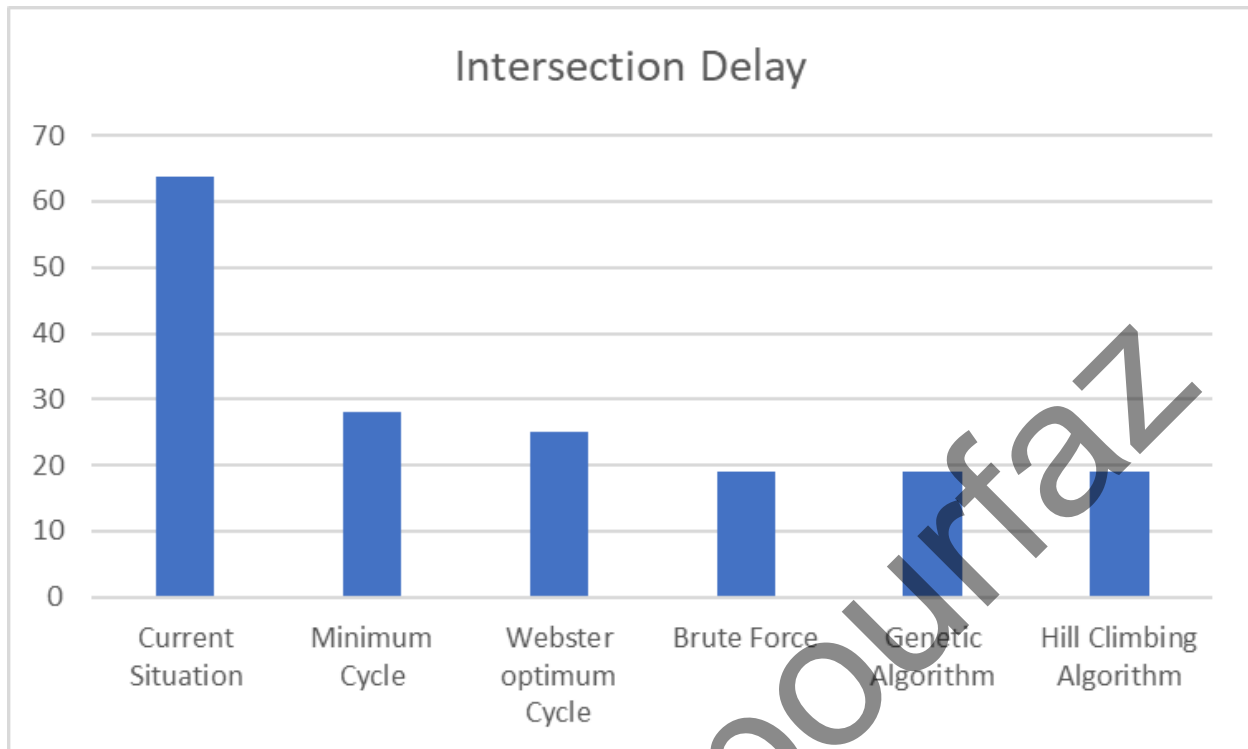


Table 7.10.1: Comparison Between Different Methods

7.4. Optimization of the 3rd Intersection

The third intersection is Via Olevano Romano - via Prenestina - via Prenestina. The current overall delay of the 3rd intersection is calculated about 21.9 [sec] and the level of service of intersection is C.

The first step is to consider the capacity constraints to obtain the minimum cycle length to serve all the demand and its respective green duration; this is reported in Table together with the optimum cycle for Webster. The delay for the Minimum Cycle is 11.33 (sec), and for Webster Optimum Cycle is 20.3 (sec).

By implementing the minimization of the delay problem in the MATLAB, we have the opportunity of investigating cycle length and green ratio variation simultaneously. but it should be in the boundaries.

By having total lost time and the flow of lane grouping from the first part of the project we can calculate Cmin. This is the lowest possible value for the minimum cycle and upper bound for cycle variation we used a value for example 5 experimentally because you have explained that in the real world we don't have a cycle like for example 10 min. so by defining a loop over the cycle length for each step we can have an internal loop that is responsible for green ratio variation.

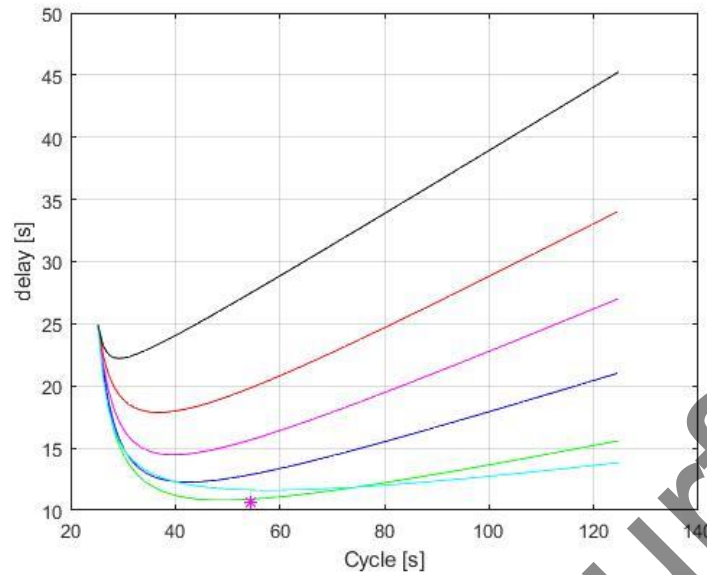


Figure 7.11: Variation of delay with respect to C keeping G1/G2 unvaried

In Figure The best solution, corresponding to a delay of 10.6 seconds, refers to a cycle length of 53.87 seconds. Each color represents a different green ratio in the figure.

Green ratio

for determining the lower and upper boundary of the green ratio the variation we need to derive the formula that links cycle length and green split. Based on this formula:

$\frac{g1}{g2} = \frac{y1}{y2}$ and if we have maximum for $y2$, we will have a minimum value for the green

split. So, by $C_{min} = \frac{L}{1-y}$, we can derive $y2$ which means the critical value of the second phase. In the same way for having a maximum green split, we derive $y1$ based on the minimum cycle to have the maximum number in numerator.

So:

Minimum value for $(g1/g2) = y1/(1 - y1 - L/C)$

Minimum value for $(g1/g2) = (1 - y2 - L/C)/y2$

in this way, we can calculate the minimum delay by considering cycle length and green ratio simultaneously.

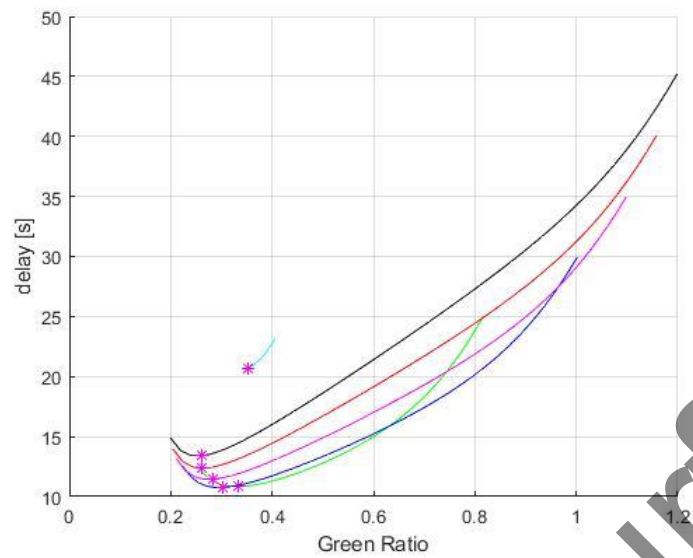


Figure 7.12: Variation of delay with variation of Green ratio

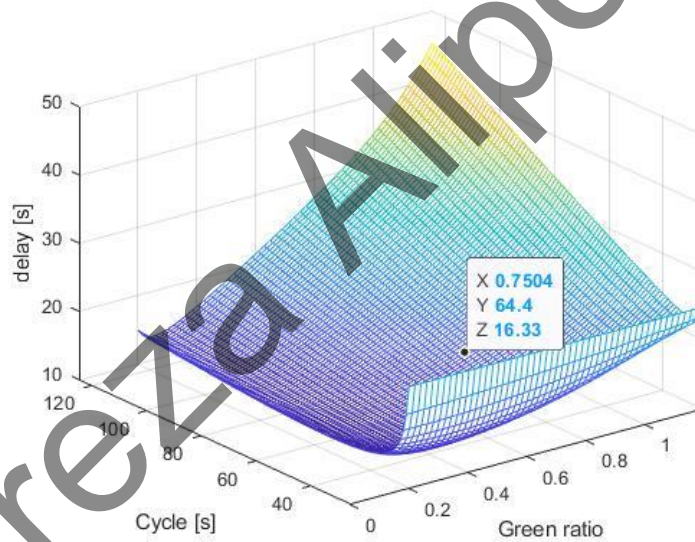


Figure 7.13: Variation of delay with variation of Green ratio and Cycle time

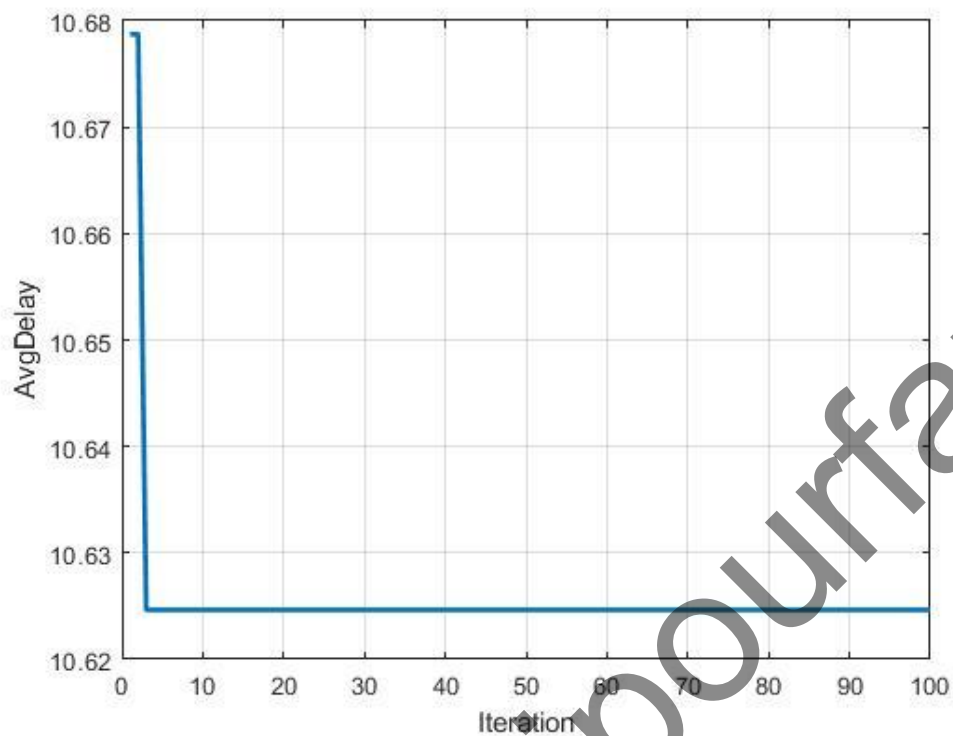


Figure 7.9: Evolution of the minimum Delay for every generation (Genetic Algorithm)

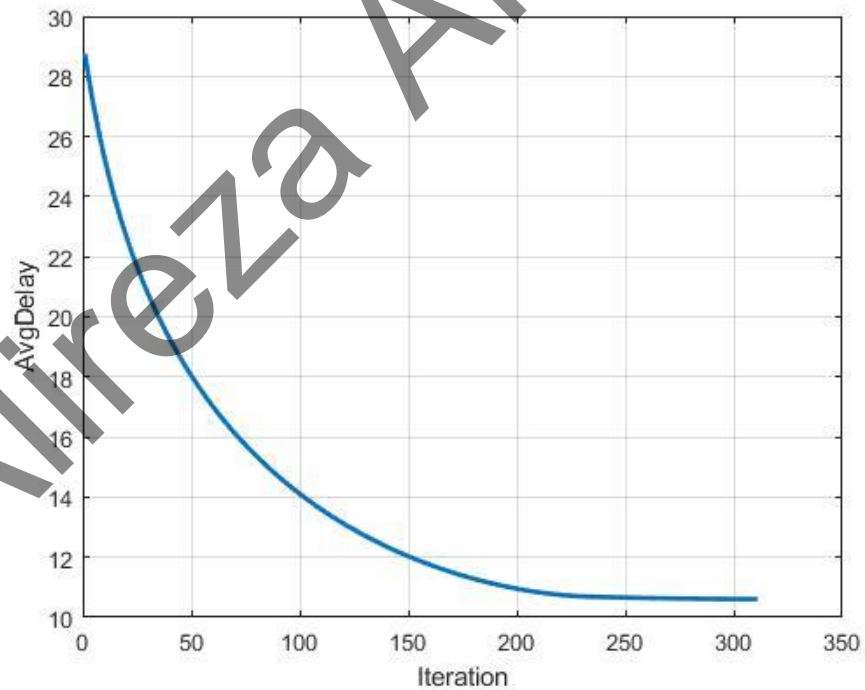


Figure 7.10: Evolution of the minimum Delay for every generation (Hill Climbing Algorithm)

The 2 figures in previous slide presents optimized delay at every iteration corresponding green splits and cycle length for the intersection. Green splits and cycle length were the input decision variables for both algorithms and were programmed and controlled to have values in the boundaries. The objective was to select the best possible combination of green splits and cycle length in response approaching traffic volume to minimize the objective function i.e., average vehicle delay at the intersection.

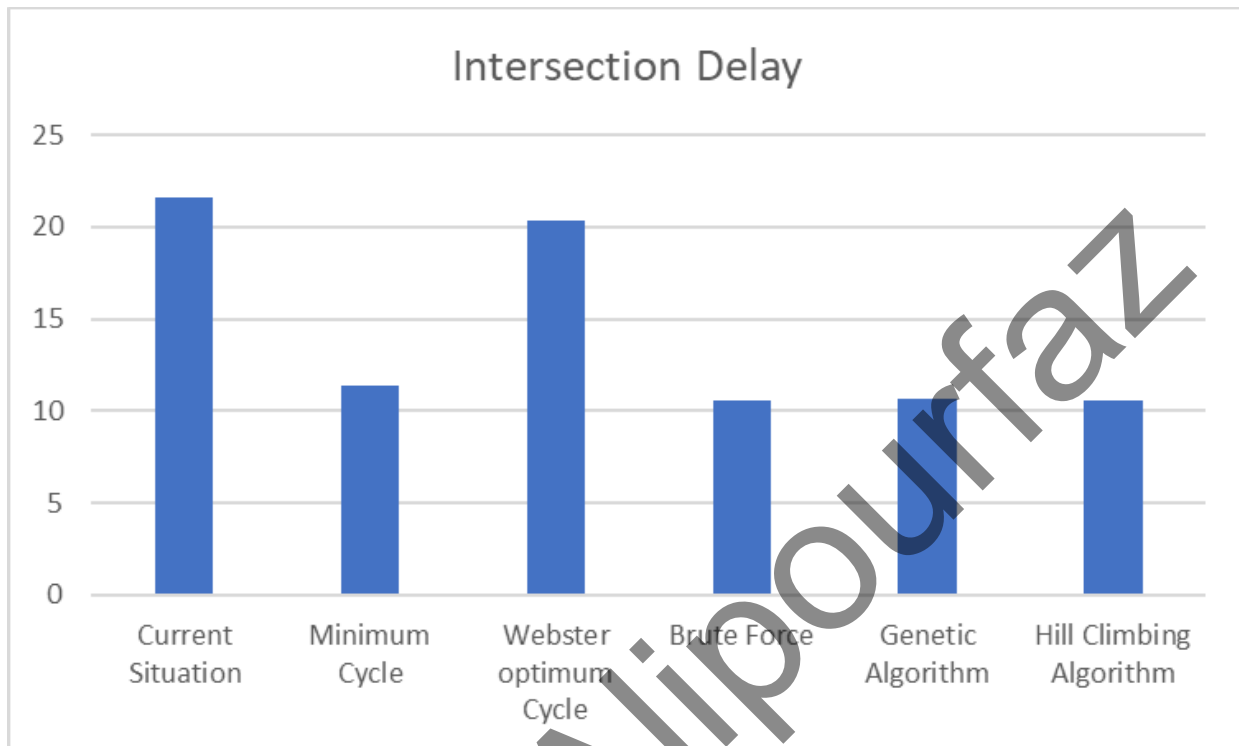
	Current Situation	Minimum Cycle	Webster optimum Cycle	Brute Force	Genetic Algorithm	Hill Climbing Algorithm
Cycle	128	25.09	49	53.73	51.06	53.83
Intersection Delay	21.56	11.33	20.3	10.6	10.62	10.6
LOS	C	B	C	B	B	B
g1	75	10.34	10.1	32.5	30.34	32.67
g2	45	3.75	2.8	10.15	9.72	10.15

Table 7.11: Comparison of different methods - 3rd intersection

	Current Situation	Minimum Cycle	Webster optimum Cycle	Brute Force	Genetic Algorithm	Hill Climbing Algorithm
Run time (sec)	-	-	-	120	2.7	0.8
Processor	Intel® Core™ i7-5500 U CPU @ 2.4GHz (4CPUs)					
Memory	8 GB					
Graphic Card	AMD Radeon R7 M265 Series					

Table 7.12: Comparison of different methods - 3rd intersection

Table 7.12.1: Comparison Between Different Methods



8. Synchronization

After the analysis of single junctions, the artery can be studied. The goal is to minimize the delay of the entire artery.

Each intersection is affected by the departure and the flow from the close upstream intersection. The reason of a signal's coordination (or synchronization), which regulates the starting of green at intersections, is to allow vehicles arriving at the downstream intersection during the green phase, in order to reduce the delay.

A synchronization problem takes as given:

- the distance between consecutive intersections l_i
- the signal setting at each intersection g_i, C_i (in order to have a periodical repetition of coordination, the cycle length has to be chosen equal for all the intersections)
- the synchronization speed in the two directions, assuming that vehicles travel along the artery at that speed, v_1, v_2

and gives as output a vector of the signals' offsets θ_{ij} , which minimizes the delay of vehicles travelling along the artery.

In the ideal synchronization:

- all the intersections are equally spaced $l_{ij}=A \forall i \neq j \ i=1, 2, \dots, n$
- there is no entering or exiting flow along the artery, it means that the green and the cycle are the same for each intersection.
- traffic flow is lower than a given value

Under the condition of constant traffic flow, the green will be the same for all the intersections and so also the cycle will be unique for the artery. In this way all the vehicles

will move at the same constant speed, forming a compact platoon and bands of vehicles' trajectory along the artery can be individuated. A vector of offsets, which make the delay at intersections nil, can be found: at each node the green will start as soon as the first vehicle of the platoon arrives and will end as soon as the last vehicles passes, such that it is possible to have a platoon moving within bands without being stopped.

This can be expressed by imposing the offset (θ_{ij}) equals the running time between two consecutive nodes:

$$t_{ij} + t_{ji} = \frac{A}{v_1} + \frac{A}{v_2} = mC$$

where m is an integer

If $v_1 = v_2 = v$ it can be obtained:

$$A = \frac{mvC}{2}$$

So that the solution of the ideal synchronization is:

$\theta_{ij} = 0$ if $x_j - x_i = 2A = mC$ for $m=0,1,2, \dots$

$\theta_{ij} = C/2$ if $x_j - x_i = 2A = (2m+1)C/2$ for $m=0,1, 2, \dots$

The ideal synchronization is usually unfeasible in real cases, because:

- junctions are not equally spaced
- flow is not uniform along the artery
- green splits are not equal

These cases can be solved by two different approaches:

- Minimum delay problem

- Maximal green bandwidth problem

The problem of the minimum delay is non convex, so it doesn't have a unique solution. It can be solved by simulation programs, which reproduce the traffic conditions.

The minimum delay solution, but correlated to it because by increasing the green bandwidth (which is defined as the set of possible trajectories at constant speed that are uninterrupted along the artery) the number of vehicles within it, so not delayed, increases.

In the formulation of the problem the bandwidths in oppose directions are different and the offset can be a value between 0 and 1. By imposing the same bandwidth in the opposite directions ($b=b'$) the symmetric problem (MB1) is obtained, which is:

$$\begin{aligned}
 \max f &= (b+b') \\
 b &= b' > 0 \\
 \theta_{ij} &= 0 \text{ or } \frac{1}{2} \\
 v_{\min} &\leq v \leq v_{\max} \\
 \max \{C_{\min, i}\} &\leq C \leq C_{\max} \\
 C_{\min, i} &= \frac{L_i}{(1 - \max_h \{y_{i,h}\} - \max_k \{y_{i,k}\})} \\
 \max_h \{y_{i,h}\} &\leq g_i \leq 1 - \frac{L_i}{C} - \max_k \{y_{i,k}\}
 \end{aligned}$$

Where:

L_i = lost time of node i

$y_{i, h}$ = saturation degree of approach h

$y_{i, k}$ = saturation degree of approach k

$b=b'$ = bandwidth inbound and outbound

g_i = green at artery approach i

The problem can be solved using an algorithm based on Equivalent system properties (Papola, Fusco, 1998).

For any set of signals an ideal system exists (in which the distance between consecutive nodes is A and the bandwidth equals the duration of green) and has the same solution.

First of all, cycle length, green splits and synchronization speed have to be fixed. The common cycle length can be the minimum or the optimum for the most critical intersection. Green splits can be chosen follow the criterion of the equisaturation or optimizing the intersections' delay.

The first pair of intersections is the starting point: depending on the distance between them (l_{ij}), the bandwidth (b_{ij}) and the offset (ϑ_{ij}) can be evaluated.

$$\text{If } l_{ij} < \frac{A}{2} \text{ or } l_{ij} > \frac{3A}{2}$$

$$\vartheta_{ij} = 0$$

$$b_{ij} = b'_{ij} = \frac{1}{2}(g_i + g_j - \frac{l_{ij}}{A})$$

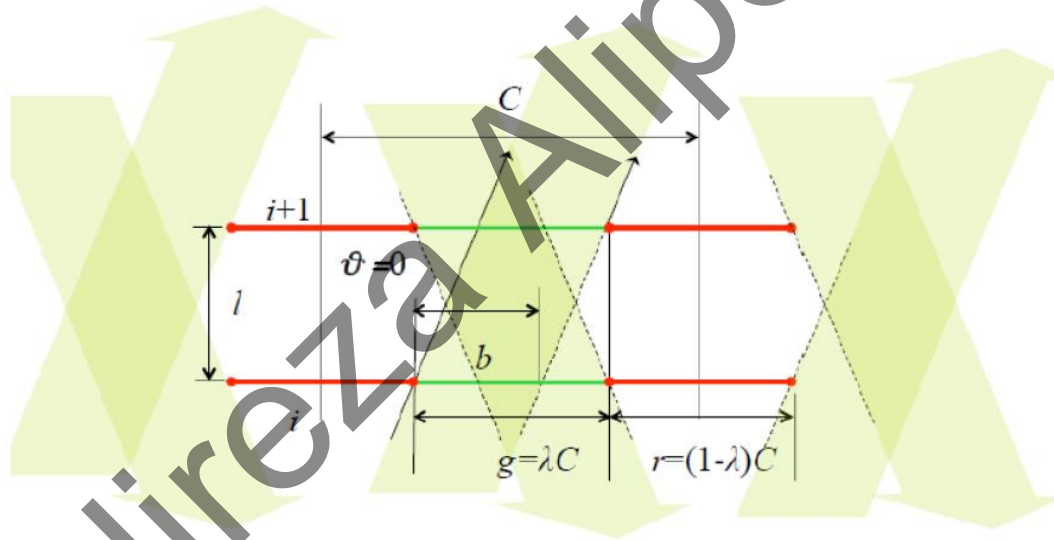


Figure 8.1: Junctions in phase

$$\text{If } \frac{A}{2} < l_{ij} < \frac{3A}{2}$$

$$\vartheta_{ij} = \frac{1}{2}$$

$$b_{ij} = b'_{ij} = \frac{1}{2}(g_i + g_j - 1 + \frac{l_{ij}}{A})$$

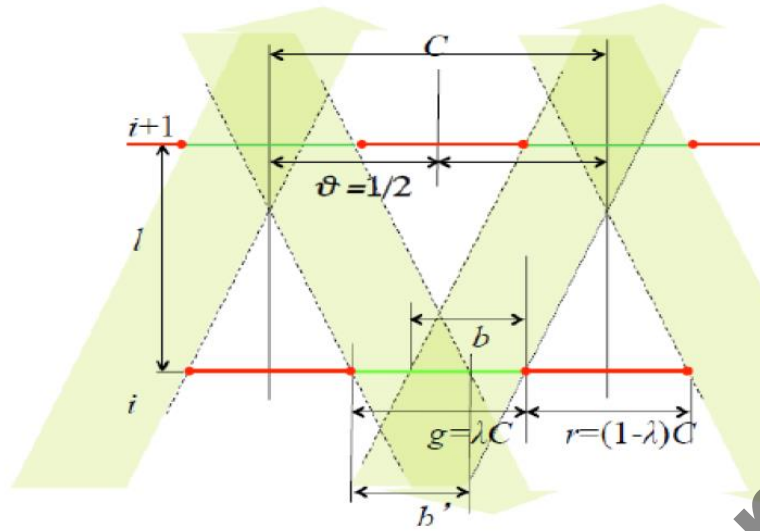


Figure 8.2: Junctions in phase opposition

Then it is necessary to find the position of the ideal node, which is equivalent to the first pair of signals. If the distance between the two nodes is less than $\frac{1}{2}$, the ideal node is between them, otherwise there are two ideal nodes, one under the first node and the other up the second node.

$$x_0 = x_i + (g_i - b_{ij})A \quad \text{if } \frac{x_j - x_i}{A} \leq 0.5$$

$$x_0 = x_i - (g_i - b_{ij})A \quad \text{if } \frac{x_j - x_i}{A} > 0.5$$

For successive nodes, which can be distant more than A , the operator mantissa is used, in order to have a distance in the range $0,1$. The successive node r is synchronized with the equivalent system corresponding to the previous nodes i, j , so that:

$$b_{ij,r} = b'_{ij,r} = \frac{1}{2} \left(g_r + b_{ij} - \text{man} \left[\frac{x_r - x_0(i,j)}{A} \right] \right) \quad \text{if } 0 \leq \text{man} \left[\frac{x_r - x_0(i,j)}{A} \right] < 0.5$$

$$b_{ij,r} = b'_{ij,r} = \frac{1}{2} \left(g_r + b_{ij} - 1 + \text{man} \left[\frac{x_r - x_0(i,j)}{A} \right] \right) \quad \text{if } 0.5 \leq \text{man} \left[\frac{x_r - x_0(i,j)}{A} \right] < 1$$

The new ideal system equivalent to the already coordinated signals is individuated.

The iterations continue until all the nodes are coordinated.

Each node, which reduces the bandwidth, shift the ideal grid, so it's necessary to evaluate the offsets after that all the nodes have been coordinated, such th

$$\vartheta = 0 \quad \text{if } 0 \leq \text{man} \left[\frac{x(i) - x_0}{2A} \right] < 0.25 \cup 0.75 \leq \text{man} \left[\frac{x(i) - x_0}{2A} \right] < 1$$

$$\vartheta = 0.5 \quad \text{if } 0.25 \leq \text{man} \left[\frac{x(i) - x_0}{2A} \right] < 0.75$$

8.1. Study Case

The next step is the synchronization of the artery. A unique cycle is necessary for all the intersections. The Maximum cycle length is chosen among all cycle length of the iterative method optimum cycle lengths, for this reason the 91.31 [sec] which is the maximum cycle length is chosen and then the Brute force method is applied one more time for find the best green ratio for each of six intersection. These have not negligible flows, for that the optimization has be done also for fixed cycle equal 91.31 [sec], in order to have the optimum green split at each intersection, so that, at least, the total delay of each of them don't increase. The details are reported in Table 8.1.

Synchronization						
Intersection	Cycle(sec)	g1(sec)	g2(sec)	g3(sec)	Delay(sec/veh)	LOS
1	91.31	33.03	23.19	19.38	33.18	C
2	91.31	65.09	15.31	-	19.19	B
3	91.31	63.24	17.1	-	11.73	B

Table 8.1: The LOS of the scenario for all the intersections

The common cycle has been chosen equal to 91.31 [sec]. The green splits have been chosen in order to have the minimum delay for each intersection with the fixed cycle. The synchronization speed has been assumed equals to 10 m/s (36 km/h), which feasible for urban area.

Intersection	Distance from previous intersection(m)	Progressive distance(m)	Green(s)	g/c
1	0	0	33.03	0.36
2	120	120	65.09	0.71
3	320	440	63.24	0.69

Table 8.2: Synchronization Data for EB

Intersection	Distance from previous intersection(m)	Progressive distance(m)	Green(s)	g/c
1	0	0	56.22	0.69
2	320	320	65.09	0.71
3	120	440	63.24	0.61

Table 8.2.1: Synchronization Data for WB

Now the A is calculated with respect to the previous assumptions, so A is equal to 456 [m].

Node		Distance	Progressive Distance	A/2 [m]	bandwidth
1	2	120	120	228	0.4
2	3	320	440	228	0.48

Table 8.3: Synchronization of the Whole Artery for EB

Node		Distance	Progressive Distance	A/2 [m]	bandwidth
1	2	320	320	228	0.514
2	3	120	440	228	0.527

Table 8.3.1: Synchronization of the Whole Artery for WB

In Table 8.3 all the bandwidth and abscissa of the ideal nodes are calculated.

Nodes			
	1	2	3
Offset	0	0	0.5
Bandwidth	$0.4 \times 91.31 = 36.52$		

Table 8.4: Final Results of Synchronization for EB

Nodes			
	1	2	3
Offset	0	0.5	0.5
Bandwidth	$0.514 \times 91.31 = 46.93$		

Table 8.4: Final Results of Synchronization for WB

In **Table 8.4** the offset and the bandwidth of the whole artery is calculated.

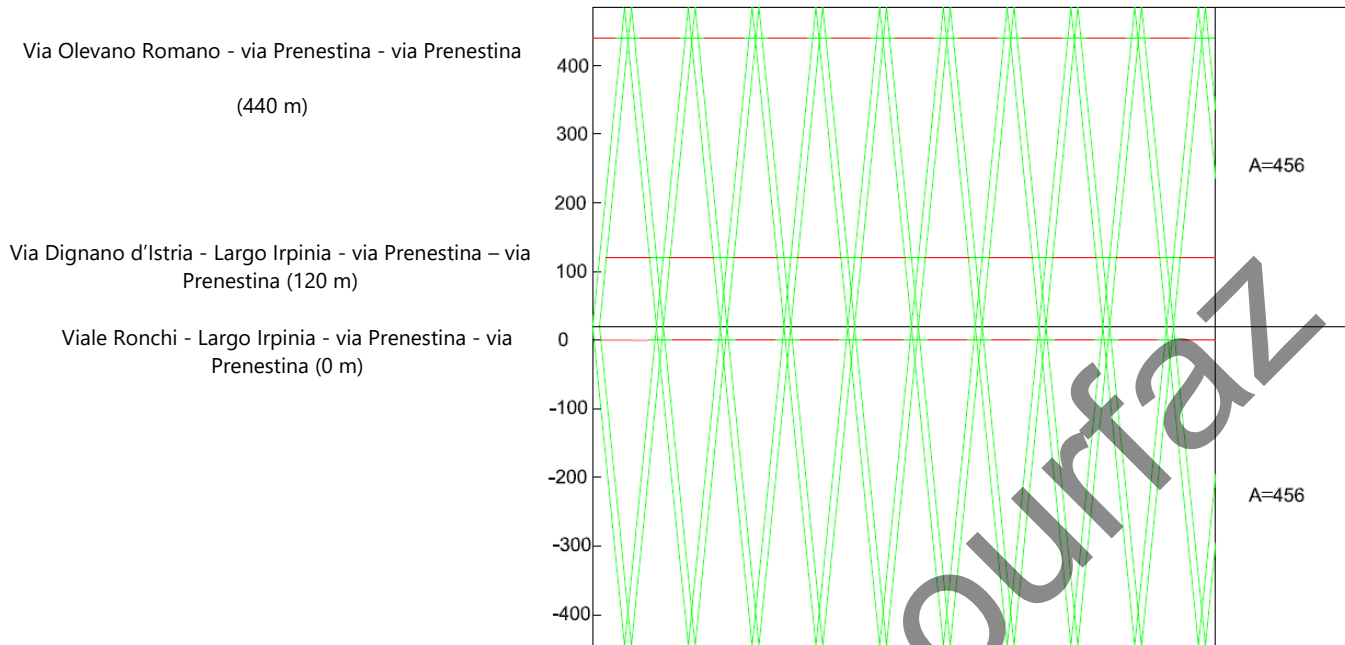


Figure 8.2.1: Synchronization of the whole artery EB

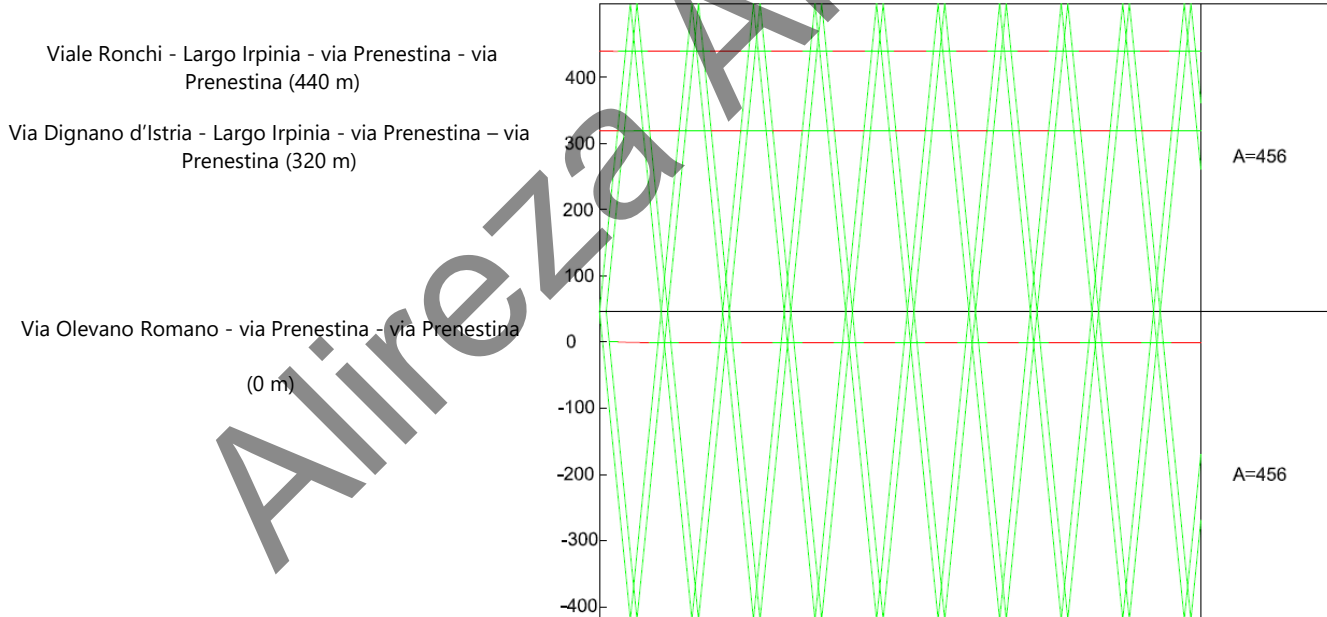


Figure 8.2.2: Synchronization of the whole artery WB

9. Conclusion

All the analyzed intersection can be improved, as single junctions, changing the signal setting, The level of service of the 1st intersection is improved from level D to level C, and finally the 2nd intersection has improved from level E to level B, and the LOS of 3rd intersection is improved from level C to level B,

The synchronization of the whole artery guarantees a good compromise between the LOS of each single intersection.

In conclusion, we present the performance comparison of heuristic algorithms for minimization of delay at intersections. These algorithms are compared with each other based on some parameters like run time, step size, and the output. For the heuristics studied in this paper, overall, Brute Force performs better than heuristics but it takes more time. When we do not have big data like this paper, I recommend to use Brute Force Algorithm because the result would be more accurate. While heuristic algorithm is designed to solve a problem in a faster and more efficient fashion than traditional methods by sacrificing **optimality, accuracy, precision, or completeness** for speed. Heuristic algorithms often times used to solve NP-complete problems, a class of decision problems.

Appendix A

```

% Author: Alireza Alipourfaz
% alipourfaz.1898332@studenti.uniroma1.it
%% Starting point, clear everything in matlab
tic; %Start the timer
clear all;
close all;
clc;

%% Problem Formulation

FitnessFunction=@(C,g,Q,S,T) TDi(C,g,Q,S,T); % FitnessFunction
T=0.25; % [hour] %Time Interval of
analysis
L=10.9; % Lost Time
nphase=2; % Number of phases
nIntersections=1; % Number of Intersections
(static as 1 intersection)

VarSize=[1 nIntersections*nphase+1]; % Decision Chromosome genes
based on number of Intersections

nGL=3; % Number of group Lanes
Q=[822,2187,662]; % [veh/h] %Flows for lane groups
EB, WB, NB, SB
S=[2921,3454,4652]; % [veh/h] %Saturation flow
corresponding lane groups
y1=[0.633]; % [adim] %Saturation degree of
the critical movement for phase 1
y3=[0.142]; % [adim] %Saturation degree of the
critical movement for phase 2
Y=(y1+y3); % [adim] %Total saturation degree
Cm=L/(1-Y); % [s] %Minimum Cycle length
Cyclemin=Cm; % Lower bound of CYCLE
Cyclemax=5*Cm; % Upper bound of CYCLE
%% Genetic Algorithm Parameters

Maxit=100; % Max iteration
nPop=50; % Population Size

pc=0.8; % Crossover Percentage
nc=2*round(pc*nPop/2); % Number of Offsprings
(parents)

pm=0.2; % Mutation Percentage
nm=round(pm*nPop); % Number of Mutants
mu=0.1; % Mutation Rate

pe=0.05; % Elite percentage

```

```

ne=round(pe*nPop); % Number of Elites

beta=8; % Selection Pressure

%% Initialization

% Individual Structure
empty_individual.Green=[];
empty_individual.Cycle=[];
empty_individual.TotalDelay=[];

% Population Structure
pop= repmat(empty_individual,nPop,1);

% Initialize Population
i=1;

while i<=nPop

    % Initialize Individual
    pop(i).Cycle=Cyclemin+(Cyclemax-Cyclemin)*rand; % Generate random Cycle
    greenMin1=y1*pop(i).Cycle; % lower bound of GREEN
    LIGHT 1
    greenMin3=y3*pop(i).Cycle; % lower bound of GREEN
    LIGHT 2
    greenMax1=pop(i).Cycle-L-greenMin3; % Upper bound of GREEN
    LIGHT 1
    pop(i).Green(1)=greenMin1+(greenMax1-greenMin1)*rand; %Generate radom
    Green light 1
    pop(i).Green(2)=pop(i).Green(1);
    pop(i).Green(3)=pop(i).Cycle-L-pop(i).Green(1); %Calculate Green light
    2

    if( pop(i).Green(3) < greenMin3 )
        continue;
    end

    % Individual Evaluation from Fitness Function
    for j=1:nGL
        % Measure Delay for each traffic light with current congestion
        pop(i).TotalDelay(j)=FitnessFunction(pop(i).Cycle,pop(i).Green(j),Q(j),S(j),T
    );
    end
    % Summation of Total Delays quotients
    pop(i).TotalDelay= sum(Q.*pop(i).TotalDelay)/sum(Q);
    i=i+1;
end

% Sort Population
TotalDelay=[pop.TotalDelay];
[TotalDelay, SortOrder]=sort(TotalDelay);
pop=pop(SortOrder);

% Store Best Solution

```

```

BestSol=pop(1);

% Store Best Fitness
BestDelay=pop(1).TotalDelay;

% Worst Fitness
WorstDelay=pop(end).TotalDelay;

disp(['FIRST Population.....Best TotalDelay = ' num2str(BestDelay)]);
fprintf('\n')
disp('Green Timings in seconds:');
disp([' Phase 1 Green Time = ' num2str(BestSol.Green(1))]);
fprintf('\n')
disp([' Phase 2 Green Time = ' num2str(BestSol.Green(3))]);
fprintf('\n')

%% Loop For Number of Iterations
count=0;
for it=1:Maxit

    % Calculate Selection Probabilities
    P=(TotalDelay/sum(TotalDelay));
    P=exp(-beta*TotalDelay/WorstDelay);
    P=P/sum(P);

    %% Crossover
    popc= repmat(empty_individual,nc/2,2);
    k=1;
    while k<=nc/2

        % Select Parents Indices from roulette wheel
        i1=RouletteWheelSelection(P);
        i2=RouletteWheelSelection(P);

        % Select Parents
        p1=pop(i1);
        p2=pop(i2);

        popc(k,1).Green=p1.Green;
        popc(k,2).Green=p2.Green;
        popc(k,1).Cycle=p1.Cycle;
        popc(k,2).Cycle=p2.Cycle;
        popc(k,1).TotalDelay=p1.TotalDelay;
        popc(k,2).TotalDelay=p2.TotalDelay;
        % Select random crossover point
        i=randi([1 3]);

        % crossover randomness
        if(i==1)

            popc1=popc(k,1).Green(3);
            popc(k,1).Green(3)= popc(k,2).Green(3);
            popc(k,2).Green(3)= popc1;

```

```

    popc(k,1).Green(1)= popc(k,1).Cycle-L-popc(k,1).Green(2);
    popc(k,2).Green(1)= popc(k,2).Cycle-L-popc(k,2).Green(2);

    popc(k,1).Green(2)= popc(k,1).Green(1);
    popc(k,2).Green(2)= popc(k,2).Green(1);

elseif(i==2)

    popc1=popc(k,1).Green(1);
    popc(k,1).Green(1)= popc(k,2).Green(1);
    popc(k,2).Green(1)= popc1;

    popc(k,1).Green(3)= popc(k,1).Cycle-L-popc(k,1).Green(1);
    popc(k,2).Green(3)= popc(k,2).Cycle-L-popc(k,2).Green(1);

    popc(k,1).Green(2)= popc(k,1).Green(1);
    popc(k,2).Green(2)= popc(k,2).Green(1);

else

    popc1 = popc(k,1).Cycle;
    popc(k,1).Cycle = popc(k,2).Cycle;
    popc(k,2).Cycle = popc1;

    popc(k,1).Green(1)= popc(k,1).Cycle-L-popc(k,1).Green(3);
    popc(k,2).Green(1)= popc(k,2).Cycle-L-popc(k,2).Green(3);

    popc(k,1).Green(2)= popc(k,1).Green(1);
    popc(k,2).Green(2)= popc(k,2).Green(1);

end

% check if new green times are out constraints.

if( popc(k,1).Green(1)<y1*popc(k,1).Cycle ||
popc(k,1).Green(3)<y3*popc(k,1).Cycle)
    continue;
end
if( popc(k,2).Green(1)<y1*popc(k,2).Cycle ||
popc(k,2).Green(3)<y3*popc(k,2).Cycle)
    continue;
end

% Evaluate Generated Offsprings for each traffic light according to
% the corresponding traffic congestion
for j=1:nGL

popc(k,1).TotalDelay(j)=FitnessFunction(popc(k,1).Cycle,popc(k,1).Green(j),Q(
j),S(j),T);

popc(k,2).TotalDelay(j)=FitnessFunction(popc(k,2).Cycle,popc(k,2).Green(j),Q(
j),S(j),T);
end

```

```

% TOTAL DELAY which corresponds to the 4 lane groups
popc(k,1).TotalDelay= sum(Q.*popc(k,1).TotalDelay)/sum(Q);
popc(k,2).TotalDelay= sum(Q.*popc(k,2).TotalDelay)/sum(Q);

k=k+1; %step
end

% Make 2 columns into 1
popc=popc(:);
% Sort popc matrix according to TotalDelay
TotalDelay=[popc.TotalDelay];
[TotalDelay, SortOrder]=sort(TotalDelay);
popc=popc(SortOrder);

%% Mutation
% Create empty Matrix with length the number of mutants
popm= repmat(empty_individual,nm,1);
k=1;
while k<=nm

    % Select Parent population
    i=randi([1 nPop]);
    p=pop(i);

    % Apply Mutation
    nVar=2;
    nmu=ceil(mu*nVar);

    j=randi([1 nVar]);
    prosimo=randi([-1 1]);
    sigma=prosimo*pm*(pop(i).Cycle);

    mutated=p.Green(j)+sigma;

    popm(k).Green = p.Green;
    popm(k).Green(j)=mutated;
    popm(k).Cycle = p.Cycle;

    if j==1
        popm(k).Green(3)=popm(k).Cycle-L-popm(k).Green(1);
        popm(k).Green(2)= popm(k).Green(1);
    else
        popm(k).Green(1)= popm(k).Cycle-L-popm(k).Green(3);
        popm(k).Green(2)= popm(k).Green(1);
    end

    if( popm(k).Green(1)<y1*popm(k).Cycle ||
popm(k).Green(3)<y3*popm(k).Cycle)
        continue;
    end

    for j=1:nGL

```

```

        % Evaluate Mutant

popm(k).TotalDelay(j)=FitnessFunction(popm(k).Cycle,popm(k).Green(j),Q(j),S(j),T);
    end
    % Summation of delay quotients
    popm(k).TotalDelay= sum(Q.*popm(k).TotalDelay)/sum(Q);
    k=k+1; %step
end
%% Merge Population

pop=[pop
      popc
      popm]; %#ok

% Sort New Population according to TotalDelay
TotalDelay=[pop.TotalDelay];
[TotalDelay, SortOrder]=sort(TotalDelay);
pop=pop(SortOrder);

% Update Worst Cost
WorstDelay=max(WorstDelay,pop(end).TotalDelay);

% Keep the Best Population from the given number
pop=pop(1:nPop);
TotalDelay=TotalDelay(1:nPop);

% Store Best Solution Ever Found
BestSol=pop(1);

% Store Best Cost Ever Found
BestDelay(it)=BestSol.TotalDelay;

% Show Iteration Information
disp(['Iteration ' num2str(it) ': Best TotalDelay = '
num2str(BestDelay(it))]);
fprintf('\n')
disp('Green Timings:');
fprintf('\n')
disp(['Phase 1 Green Time = ' num2str(BestSol.Green(1)) ' seconds']);
fprintf('\n')
disp(['Phase 2 Green Time = ' num2str(BestSol.Green(3)) ' seconds']);
fprintf('\n')

%end of generation
it=it+1;

end

%% Results / Plots
figure(1);
plot(BestDelay,'LineWidth',2);
% plot(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('AvgDelay');

```

```
grid on;  
toc %stop the timer
```

Roulette Wheel Selection Function

```
function i=RouletteWheelSelection(P)  
  
    r=rand;  
  
    c=cumsum(P);  
  
    i=find(r<=c,1,'first');  
  
end
```

Fitness Function

```
function avgDelay=TDi(C,g,Q,S,T)  
  
    x = (Q.*C)./(S.*g);  
    d1=(.5*C*((1-(g/C)).^2))./(1-(min(1,x).*(g/C)));  
    c=g.*S./C;  
    d2 = (900*T*((x-1)+sqrt(((x-1).^2)+((4*x)./(c*T)))));  
  
    avgDelay=(d1+d2);  
  
end
```

Appendix B

```
% Author: Alireza Alipourfaz
% alipourfaz.1898332@studenti.uniroma1.it

clear all
close all
clc
tic % Start the timer

L=10.9; %[s] %Lost time [s]: Sum lost times for critical movements of
different signal phases
T=0.25; %[hour] %Time Interval of analysis
Q=[822,2187,662]; %[veh/h] %Flows for lane groups
S=[2921,3454,4652]; %[veh/h] %Saturation flow for lane groups
y1=[0.633]; %[adim] %Saturation degree of the critical movement for phase 1
y2=[0.142]; %[adim] %Saturation degree of the critical movement for phase 2
Y=(y1+y2); %[adim] %Total saturation degree
Cm=L/(1-Y); %[s] %Minimum Cycle length
Cyclemin=Cm; % Lower bound of CYCLE
Cyclemax=5*Cm; % Upper bound of CYCLE

initial=[]; % initial values of cycle length and green ratio
initial(1)= Cyclemin+(Cyclemax-Cyclemin)*rand; %Generate random number for
Cycle
ulmin=y1/(1-y1-L/initial(1)); %[adim] %Determine the lowest value for the
green ratio g1/g2 (notice that it depends on the cycle length value)
ulmax=(1-y2-L/initial(1))/y2; %[adim] %Determine the highest value for the
green ratio g1/g2 (notice that it depends on the cycle length value)
initial(2)=ulmin+(ulmax-ulmin)*rand; %Generate random number for green ratio
g1=initial(2)./(1+initial(2)).*(initial(1)-L); %Calculating the green of
phase 1
g2=initial(1)-L-g1; %Calculating the green of phase 2
G=[g1 g1 g2]; % Define a vector of green corresponding to group lanes
%Calculating the delay for initial values
for i=1:3
iDelay(i) = objectiveFunction(initial(1), G(i),Q(i),S(i),T);
end
initialDelay=sum(Q.*iDelay)/sum(Q); %initial Delay

stepSize = [0.1 , 0.01]; %Define the size of steps

iteration = 1;
value.delay=[]; %Define a structure for storing delay of every iteration
value.greenratio=[]; %Define a structure for storing green ratio of every
iteration
value.cycle=[]; %Define a structure for storing cycle of every iteration
bestcycle=initial(1);
bestgreenratio=initial(2);
minDelay1=initialDelay; %Define variable for while function when we are
calculating neighbors of greenratio
```

```

minDelay2=initialDelay; %Define variable for while function when we are
calculating neighbors of cycle
Neighbours=zeros(2,2); %Define a vector for neighbors
improvement1 = 0;
    while improvement1 == 0

        Neighbours=[bestcycle bestgreenratio+stepSize(2)      %Defining
neighbours
                    bestcycle bestgreenratio-stepSize(2)
                    bestcycle+stepSize(1) bestgreenratio
                    bestcycle-stepSize(1) bestgreenratio
                    bestcycle+stepSize(1) bestgreenratio+stepSize(2)
                    bestcycle+stepSize(1) bestgreenratio-stepSize(2)
                    bestcycle-stepSize(1) bestgreenratio-stepSize(2)
                    bestcycle-stepSize(1) bestgreenratio+stepSize(2)];

        g1=Neighbours(:,2)./(1+Neighbours(:,2)).*(Neighbours(:,1)-L);
%Calculating the green of phase 1
        g2=Neighbours(:,1)-L-g1; %Calculating the green of phase 2
        G=[g1(1) g1(1) g2(1)
            g1(2) g1(2) g2(2)
            g1(3) g1(3) g2(3)
            g1(4) g1(4) g2(4)
            g1(5) g1(5) g2(5)
            g1(6) g1(6) g2(6)
            g1(7) g1(7) g2(7)
            g1(8) g1(8) g2(8) ];
%Calculating the delay
        for j=1:8
            for i=1:3
                iDelay(i) = objectiveFunction(Neighbours(j,1),G(j,i),Q(i),S(i),T);
            end
            value(iteration).delay(j)= sum(Q.*iDelay)/sum(Q);
            value(iteration).greenratio(j)=Neighbours(j,2);
            value(iteration).cycle(j)=Neighbours(j,1);
        end
        for k=1:8
            if (value(iteration).delay(k)<minDelay1)
                minDelay1=value(iteration).delay(k)
                bestcycle=Neighbours(k,1);
                bestgreenratio=Neighbours(k,2);
            end
        end
        if (value(iteration).delay(1)>minDelay1 &&
value(iteration).delay(2)>minDelay1 && value(iteration).delay(3)>minDelay1 &&
value(iteration).delay(4)>minDelay1 && value(iteration).delay(5)>minDelay1 &&
value(iteration).delay(6)>minDelay1 && value(iteration).delay(7)>minDelay1 &&
value(iteration).delay(8)>minDelay1)
            improvement1=1;
        end
        bestDelay(iteration)=minDelay1;
        iteration=iteration+1;
    end
%% Results / Plots
figure(1);

```

```
plot(bestDelay,'LineWidth',2);  
% plot(bestDelay,'LineWidth',2);  
xlabel('Iteration');  
ylabel('AvgDelay');  
grid on;  
toc      % Stop the timer
```

Objective Function

```
function [ d ] = objectiveFunction(C,g,Q,S,T)  
  
    x = (Q.*C)./(S.*g);  
    d1=(.5*C*((1-(g/C)).^2))./(1-(min(1,x).*(g/C)));  
    c=g.*S./C;  
    d2 = (900*T*((x-1)+sqrt(((x-1).^2)+((4*x)./(c*T)))));  
  
    d=(d1+d2);  
end
```