Smart Traffic Light Scheduling Algorithms

Jihene REZGUI, Mamadou BARRI and Reiner GAYTA
Laboratoire **R**echerche **I**nformatique **Ma**isonneuve (LRIMa)

Montreal, Canada
jrezgui@cmaisonneuve.qc.ca

Abstract— Traffic lights have and always will be necessary for the safety of the traffic on the road. However, the time cycles of these traffic lights are based on premeditated computations and not real time data. These scheduling methods can often be inefficient and therefore lead to traffic congestions. In this paper, we propose three different Smart Traffic Light Scheduling (STLS) algorithms that were tested on our Isolated Intersection Simulator (IIS) Java platform. IIS simulates real-life traffic flow on an at-grade junction with different traffic light scheduling algorithms. We integrate a heuristic to our proposed algorithms that takes into account, not only the density of the junctions like traditional schemes, but also the waiting time of the vehicles. In addition, it can also give the right of way to emergency vehicles that need immediate passage through the intersection. Throughout extensive simulations, IIS displays the efficiency of our different STLS algorithms in terms of managing the intersection and avoiding congestions. For instance, our simulations demonstrated a tremendous decrease in the average delay time when compared to regular traffic lights. In fact, cars on the intersection using our STLS algorithms performs 3 times better. On top of this, thanks to our algorithms, the average speed of the cars is nearly doubled and the number of static cars on average are halved.

Keywords—ISS; Smart Traffic Light; STLS; Average Traffic Density; Traffic Simulator; Junctions; Average Waiting Time.

I. INTRODUCTION

Since the late 1800s, traffic has been utilized in order to manage the circulation of vehicles in a safely manner. With the growing number of cars on our roads, problems such as traffic congestions and gridlocks have been increasing exponentially. In fact, in modern large cities such as Toronto and New York, congestion can add up to around 60% of extra travel time to one's commute [1]. Furthermore, congestion is a major catalyst for global warming. The more time a car spends on the road; the more CO_2 it produces. By reducing the amount of congestions, we will effectively reduce our emissions. Therefore, by increasing the average traffic speed by approximately 20 miles per hour, we reduce our CO2 emissions by 21 metric tons every month [2]. These problems demonstrate the necessity of a more efficient traffic management system. Traditional traffic lights no longer meet our needs, as they do not take into account the current state of the traffic it is trying to manage. Thus, they cannot fulfill the specific needs of every intersection.

We propose the use of Smart Traffic Lights (STL's) in the place of time-based traffic lights. STL's, unlike regular traffic lights, will be connected to a group of cameras or sensors watching over the intersection. Thanks to these cameras or sensors, the traffic lights will be able to collect real time data on the cars on the intersection and adjust the flow accordingly in order to maximize traffic flow.

Many have attempted to resolve this problem by using several algorithms that acknowledge the needs for optimal traffic flow. With our STLS algorithms, we have achieved improvements on algorithms discussed in prior researches in ways that are explained in the following section.

Our contributions in this paper can be summarized as follows: (1) We introduce our Java platform, called "IIS" [3] which simulates several intersections and compares different variations of our STLS algorithms; (2) We propose algorithms in order to increase the flow of traffic; (3) We demonstrate the effectiveness of the improvements made with STLSDT (Smart Traffic Light Scheduling based on Density and delay Time) heuristic on our IIS platform.

Section II provides a brief overview of related work and compare them with our platform. Section III displays our intersection traffic simulator (IIS). Section IV presents the different variations of our STLS algorithms. Section V shows IIS's simulation results and Section VI concludes our paper.

II. RELATED WORK

This section is divided into two sub-sections. The first one presents other attempts at the making of traffic management algorithm and the second one explains the diverse and simple ways our algorithm can be implemented in the real world.

1. Traffic management algorithms

A. Dynamic Coded Algorithm

The authors in [4] present a dynamic algorithm for the traffic management of an intersection. Similar to us, they use traffic density to schedule the traffic lights. However, one major flaw in their simulation, which is a four-way junction similar to that of Fig. 1, is that only one traffic light at a time can be green. This alone cuts circulation in half, as nothing really prevents two opposing lanes from advancing at the same time.

Fig. 1. A Four-way junction with a light for a single lane

Another weakness of their algorithm is the fact that they ignore the traffic densities of lanes with the green light. By

978-1-7281-4275-3/19/$31.00 ©2019 IEEE

doing so in a four-lane intersection, it amounts to simply diminishing the traffic light timing cycle, which has been proven to be effective in urban areas [5]. Their improvements would therefore be a result of a shorter time cycle instead of a smart traffic algorithm and their results would not apply to most of real-life situations. However, in IIS, we simulate and test intersections that are similar to today's traffic systems.

B. Intelligent Traffic Light Controlling Algorithm

In [6], the authors present a traffic management algorithm based solely on traffic density. This presents a major problem, as it is very common for four ways junctions (which is what they based their algorithm on) to be comprised of two major lanes and two minor lanes. It is therefore possible for the major lanes to constantly have a higher density than the minor lanes.

For example, if there was only one car in the minor lanes and the major lanes always had at least two cars, the major lane will always have the green light. Sure, this will provide the maximum possible flow; however, it will also result in having that one car waiting for an indefinite amount of time. To resolve this issue, we have also taken into account the total delay time of the cars on the intersection.

Another problem comes with them ignoring the types of vehicles in the intersection. Thus, unlike in our algorithm, responding emergency vehicles are not taken into account and these vehicles would have to hope that the lane they are in has the higher density of the four.

2. *Ease of Integration of STLS algorithms in existing technologies*

A. Artificial Intelligence: Computer Vision

Our simulator relies on traffic data generated by our IIS platform. However, with the multiple advances made in Artificial Intelligence, many computer vision algorithms such as YOLO [7] have been optimized and proven capable to recognize cars in video footage. For instance, many researches have shown great results considering the recognition and the counting of cars in an intersection or on a highway [8].

Results of around 95% accuracy were presented in the *Image-base Vehicle Analysis using Deep Neural Network* paper [8]. Based on these results, a possible real-life application would be installing cameras on an intersection and using them to count the cars and therefore the density of each lane. This collected data can be used in our STLS algorithm in order to manage the traffic lights.

B. VANETs in the Urban Environment

An alternative to using computer vision AI in order to count the car density and the average delay time is to use a vehicular ad-hoc network (VANET) [9] which can provide us with data of different car characteristics such as position, delay time and speed.

These VANETs can be used in order to create an Intelligent Transportation System (ITS) by taking the data, analyzing it with our STLS algorithm and sending the output scheduling to nearby traffic lights through the Dedicated Short-Range Communications (DSRC) channels. These channels have been assigned by the US Federal Communication Commission (FCC) department 75 MHz of bandwidth at 5.9 GHz and have an approximate range of 300 meters. These unique frequencies ensure that inferences with other channels are minimized.

Considering the fact that many vehicles are already integrated in the VANETs systems, it could prove to be more cost effective than installing cameras on each intersection as proposed in section *A*. VANETs can therefore be an efficient way to collect data for our STLS algorithm.

III. ISOLATED INTERSECTION SIMULATOR (IIS)

1. *Simulation results*

Vehicle parameters*:* In our simulator, cars are generated with the following properties : car action (turn left, right or go forward), car direction (north, east, south or west), vehicle image (an array of images for aesthetic purposes of the simulator) and average speed (random values that follow a certain distribution based on real life data).

The only property, which has an equiprobable set of values, is the image property. Every other one of them has a different probability distributing which is explained in the *Vehicle behavior* section.

Every car is generated with a certain rate which was determined by us based on real life data from the Montreal City Statistics Bureau [10]. In fact, we took the average appearance rate of the cars at different intersection at peak times of the day (7:30 in the morning and 18:00 in the evening). We used three different intersections for our computations: Cavendish / Saint-Jacques, Beaubien / Papineau & Notre-Dame / Sainte-Catherine. From these busy intersections, we deduced an average appearance rate of 0.788 vehicles per second. We then use Gaussian distribution to determine when a car appears. With this distribution, we are able to have varying time intervals between car appearances whilst also keeping our desired average appearance rate. 0.788 vehicles per second was the rate used for our simulations, but this parameter can be modified as well as other parameters explained in the following section.

Simulation Parameters: In order to cover most of the real-life situations, IIS enables us to modify many different parameters of the traffic flow during the simulation.

These parameters include the appearance rate of the vehicles, whether or not a certain lane has abnormal traffic, which is explained further on in this section, the average speed of the generated vehicles and the number of vehicles generated during the simulation. The appearance rate ranges from 0.2 to 2.0 cars per second. The cars can have a minimum average speed of 10km/h and a maximum of 150km/h. Finally, the maximum number of cars generated is infinity, thus it can simulate traffic

for as long as we need it to. Changing these parameters can have a drastic effect on the results. The effects of these modifications are shown in section V.

2. *Vehicle behavior*

Simulator Limitations: The physics of our simulator can be summed up as follows: Physics concepts such as friction with the tires, acceleration and decelerations are not taken into account. This choice was made because these concepts only cause a very small variation in our results and thus it is not pertinent for us to take them into account.

We assume that every car of the intersection is either static or is moving at the set average speed. This assumption is plausible because we make that assumption for both the simulation with our algorithms and the simulation that contains regular lights. Both limitations will therefore not discredit our results.

Car actions: Every generated car has one possible outcome of the three mentioned in the *A Section:* turn left, turn right or go straight forward.

Probabilities and rates: In the *Section A,* we mentioned that every car is generated with a set of parameters. All of them but the image parameter has different probabilities and rates which are implemented as follows:

A. The appearance rate

The appearance rate has a 10% probable variation from our initial value discussed in the Platform *Input Section* that is based on real life data.

B. Abnormal traffic

Abnormal traffic happens when there is an excessive rate of incoming car from one particular direction. This can happen in real life when there are multiple construction sites around the intersection thus the flow is bigger only in certain directions. Abnormal traffic affects the rate of car appearance and enables our simulator to test algorithms even when the traffic flow is not regular. To do that, we quadruple the probability of a car appearing in a certain lane.

C. Actions

In order to make our simulator realistic, the three actions of a car (turn left, turn right or go straightforward) are not equiprobable. For instance, there is a higher chance of a car going straight forward then turning left or right. In IIS, based on results from real life data [10], we decided to give cars a 30% chance to turn when they appear (15 % to turn left and 15% to turn right).

1) Turning mechanism

In order to make our vehicles turn, we make their images rotate at a certain speed relative to their movement speed until it is done turning. As for the car's movement speed, we gradually transfer its speed on one axis (x or y) to the other axis.

For example, if a car were moving towards east (which means that its speed is positive towards the x-axis) and were to turn left (which means that its speed would become negative towards the y-axis), the car's speed on the x-axis will gradually decrease until the car is aligned with the proper lane. The car's speed on the y axis would then gradually decrease (because its final speed will be negative) until its absolute value is identical to its original speed.

2) Collision avoidance:

Our first problem with collisions was with cars not being able to detect the car in front of them. To prevent cars from clipping inside of each other when one stopped; we gave each car the position of the car in front of it and made it stop when its distance to the car in front got smaller than a car length.

In order to prevent collisions whilst turning, we gave the position of oncoming vehicles to cars turning left and made these cars stop until the oncoming vehicle got out of its path.

IV. SMART TRAFFIC LIGHT SCHEDULING ALGORITHMS

1. *Traditional Traffic Lights*

Traditional traffic lights usually rely on timed cycles in order to determine when to change the traffic flow. Because of this, our simulation of an intersection with traditional lights does the same. These lights are on a shorter 51-second cycle (24 seconds green, 3 seconds yellow, 24 seconds red) which have been proven to be more effective in urban cities [5]. We gave the traditional lights as many advantages we could in order to further prove the effectiveness of our algorithms.

2. *STLSD*

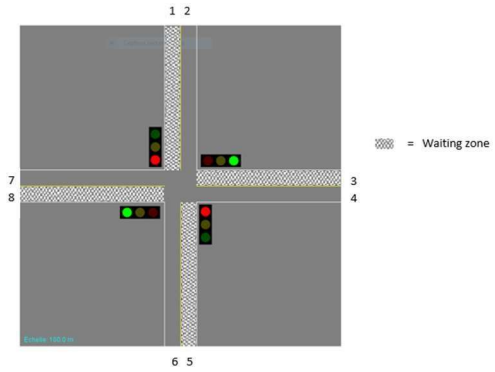

Fig. 2. An IIS empty intersection showing the waiting zones

Waiting STLSD is the variation of our STLS algorithm that is solely based on traffic density, similar to ITLC, a traffic management algorithm [6]. Our intersection is divided into 8 different lanes (as shown in figure 2), which all contain a "waiting zone".

Cars found in these "waiting zones" have yet to pass through the middle of the intersection. Depending on which lane they are on, each vehicle found inside a "waiting zone" is added to one of two lists: HorizontalDensity or VerticalDensity. These two lists are then compared, and the algorithm verifies if

the list with the higher density is the one with the green light. If so, the algorithm does nothing. If not, the algorithm changes the flow of traffic by giving the green light to the lanes that have the highest density. At the end of our calculations, the lists are wiped clean and the algorithm waits 5 seconds before recalculating the densities in order to avoid constantly changing lights.

This algorithm maximizes flow but could cause an abnormally high delay time for a vehicle that finds itself in the wrong place at the wrong time.

3. *STLSDT*

STLSDT is a variation of our algorithm that takes traffic density and the wait times of each car into account. The way we calculate the density is identical to the way we do it in STLSD. What differentiates STLSDT from STLSD is the way they manage the traffic. Similarly, this algorithm will give the green light to the lanes with the higher density.

However, if it realizes that the delay time of a car (the amount of time a car is stood still) has exceeded a certain amount of time (currently set to 10 seconds, but can be modified) and that that car still isn't moving, it will switch the traffic flow of the intersection in order to allow the waiting car to advance. This method minimizes the delay time of cars to the detriment of optimal traffic flow.

Alg.1. STLSDT algorithm

Info: **STL**: Smart Traffic Light; **WZ**: Waiting Zone; $\mathbf{d}_i$: the traffic density on the road i inside WZ; $\mathbf{t}_i$ the maximum delay time found on the road i;

```
1  compute d_i and t_i of cars in WZ;
2  while (number of cars left > 0)
3  if (t_i2 the maximum delay found on the road with the red light < 7 second) {
4          letPass(d_i2);
5  } else {
6          if (d_i1 the density of the road with the green light < d_i2 the density of the road with
   currently stopped) {
7                  letPass(d_i2);
8          }
9  }
```

4. *STLSDE (D or T)*

This algorithm is a variation of STLSD and STLSDT that takes into account emergency vehicles. It works in the same way as the previous two algorithms. However, when it detects an emergency vehicle in one of the waiting zones, it will ignore the densities and the waiting times and ensure that the emergency vehicle has the green light at all times.

When it detects two emergency vehicles in adjacent lanes (which means that they both cannot advance at the same time), the one that initially has the green light will continue to have it until it is out of the waiting zone. Once it is out of the waiting zone, the green light is given to the other lane containing an emergency vehicle.

V. SIMULATION RESULTS

1. *Simulation Input*

To obtain the results in the next subsection, the following input parameters are specified:

Our IIS simulation was executed for around 60 seconds (1 minute) depending on the car appearance rate. The dimensions of the intersection are 100 meters by 100 meters. The simulated cars are 4 meters long and 2 meters wide which are the dimensions of the average sedan [11]. In addition, 60 cars are generated for each simulation.

With these input parameters, we obtained the results stated in subsection *B*.

2. *Generated results*

A. IIS Interface

Fig. 3. IIS's Interface with 3 types of simulation of different STLS algorithms

B. Algorithms performance in different situations

To properly test our algorithms, we wanted to simulate traffic at different situations.

i) Rush-hour with normal traffic

During the rush-hour situations, we set our rate appearance to 0.789 cars / second as determined in the section *A* based on real life data.

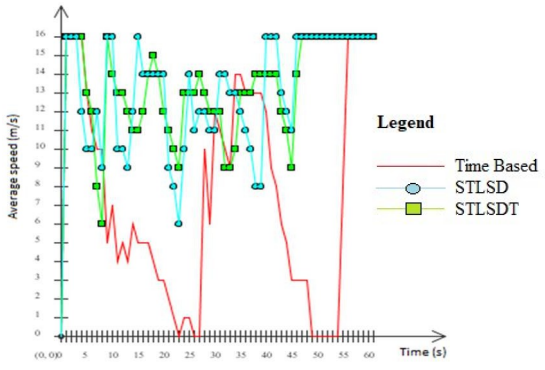

Fig. 4. The average speed (m/s) with abnormal traffic.

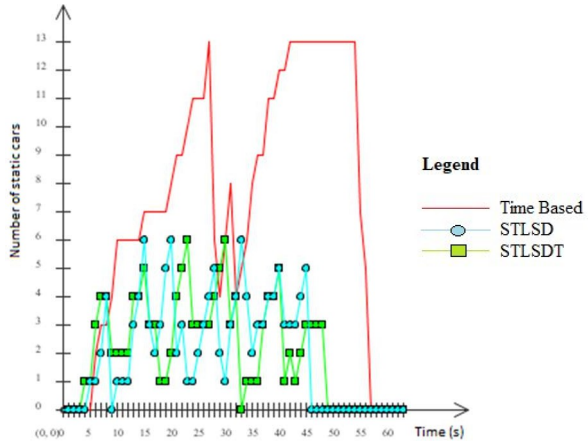

Fig. 5. The number of static cars on the intersection in real time.

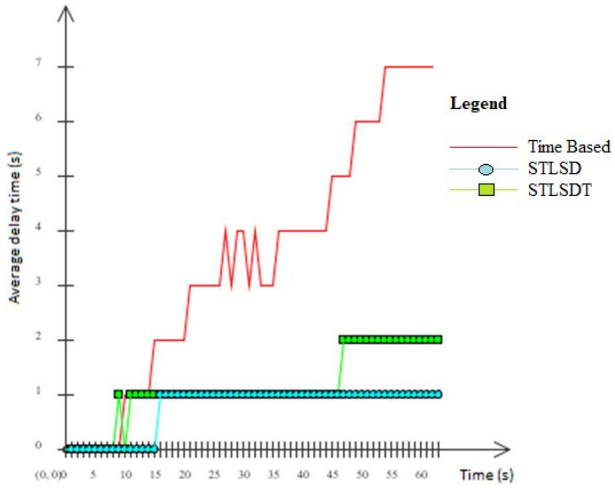

Fig. 6. The average waiting time (seconds) in real time.

In Fig.4, we see that both STLSD and STLSDT performed somewhat equally and boast far superior results than the regular scheduling system. With our algorithms, the average speeds of all cars in the intersection never reached below 5 m/s whereas the average speed for the cars in the intersection regulated by regular traffic lights is only around 7 m/s. In fact, there is even a moment in the simulation where the average speed of cars in the regular scheduling intersection reaches 0 m/s. This indicates that none of the cars in the simulation was moving. This unwanted situation never happens in both simulations running our algorithms.

In Fig.5, we can see that the regular traffic light algorithm has higher local maximums. In fact, the local maximums for our STLSD and STLSDT are around 50% less than those of the time-based algorithm. We can also observe that these two algorithms perform nearly identically. The only differences that we can spot is that the STLSDT seems to have slightly higher maximums than STLSD. This phenomenon is explained by the fact that our STLSDT gives the green light to cars that have been waiting for a long time. This behavior can sometimes hinder traffic flow.

In Fig. 6, the curves of our STLS algorithms show that with regular traffic lights, the average waiting time goes up to 7 seconds on average for every car. On the other hand, our STLSD algorithm shows a very low maximum average of 2 seconds. However, our STLSDT algorithm has an even lower average of only 1 second. As we can see, by sacrificing a bitof traffic flow, STLSDT shows a better performance than STLSD. These graphs show unusually low delay times (1s-2s) because of the fact that many cars that are counted have a delay time of zero seconds. This drastically lowers their averages.

ii) Rush-hour with abnormal traffic

When a traffic flow in a particular direction is substantially higher than the flows in other direction, the intersection is submitted to *abnormal traffic*. These situations often happen in real life when roads are closed due road woks. In order to simulate these situations, we took the horizontal lanes (east and west) and quadrupled the number of cars that would appear on them.

Compared to the *Rush Hour with normal traffic* results, our STLSDT algorithm shows better results than our STLSD algorithm, which are still both better than the normal traffic lights scheduling system. In fact, STLSDT has slightly fewer static cars, a better average speed and a lower average delay time. The following graphs show our extended testing in an *abnormal traffic* flow situation.

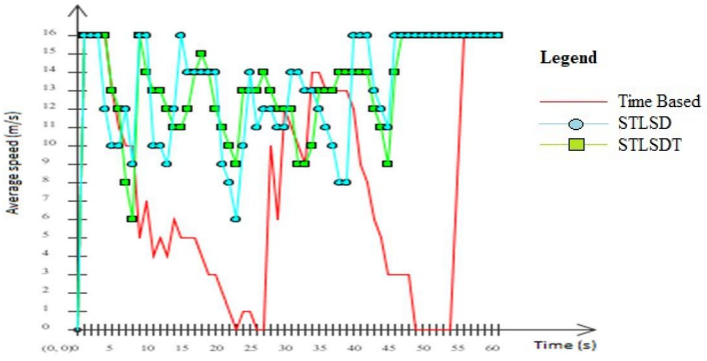

Fig. 7. The average speed (m/s) with abnormal traffic.

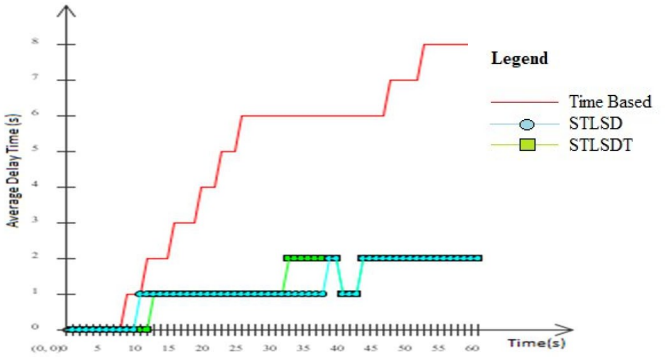

Fig. 8. The average speed delay time (s) of all the simulated cars

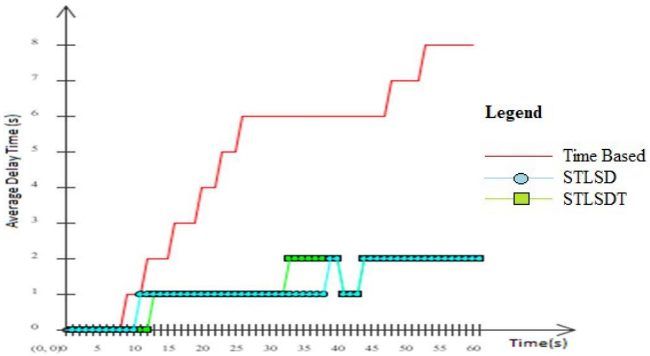

Fig. 9. The average speed delay time (s) of all the simulated cars

Fig. 7 shows us that the cars in the junction manage to reach a higher maximum speed more often than our STLSD. In Fig.8, we see that both STLSD and STLSDT have the same maximum delay time of 2 seconds.

However, STLSD reaches that maximum at an earlier time than STLSDT and, for most of the simulation, STLSDT has an identical, if not lower average delay time than STLSD.

Finally, as shown in Fig.9, we see both algorithms performing nearly identically.

iii) Rush-hour with emergency vehicles:

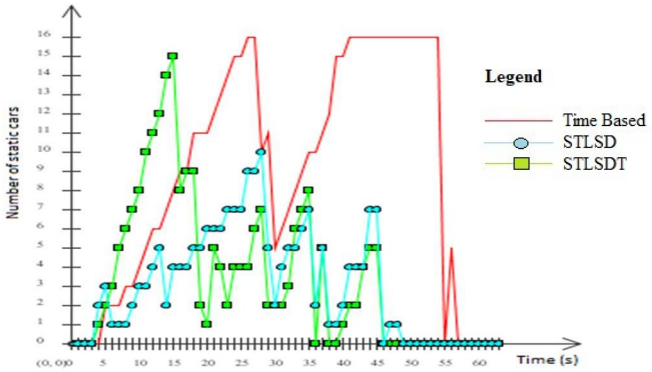

Fig. 10. The number of static cars on the intersection in real time

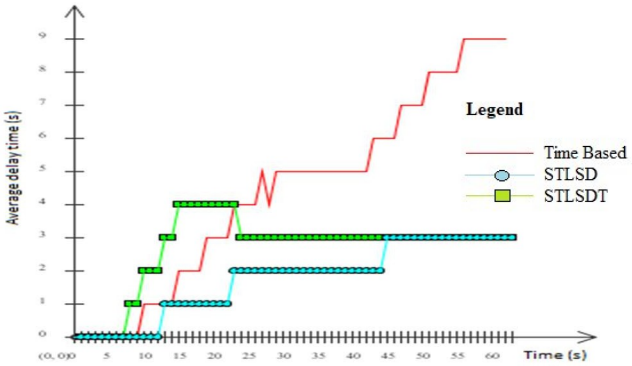

Fig. 11. The delay time (s) of all cars in real time

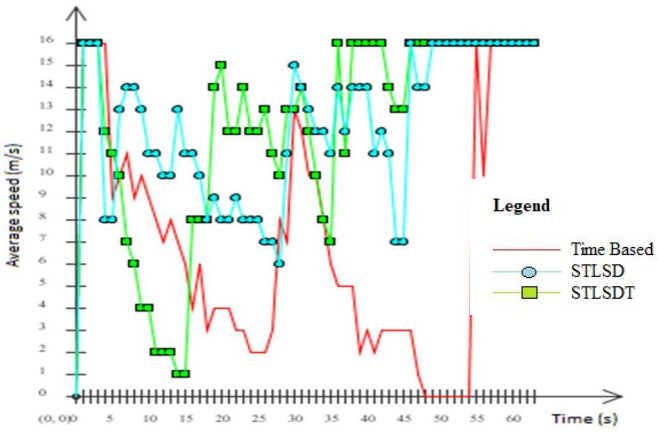

Fig. 12. The average speed (m/s) in real time.

As for the simulations that give priority to emergency vehicles, if we compare these results to our first simulations, the only major difference we can distinguish is a decrease in performance from our STLS algorithms. This is due to the fact that giving priority to emergency vehicles hinders the traffic flow. A drop in performance can be seen in both Figures 10 and 11. On all three graphs, there is a spike near the 15 second mark for our STLSD algorithm. This may suggest that this algorithm is less efficient when emergency vehicles are considered.

Figure 12 shows drastically lower average speeds for both algorithms when compared to the average speeds of situation **a)**. Our STLS algorithms have slightly lower local minimums (average local minimum of 7m/s compared to the 8m/s of situation **a)**). On top of that, they seem to reach these minimums at a much higher frequency. However, this decrease in performance is only a minor one and we believe that having a slightly less optimal traffic flow is fair price to pay in order to ensure that emergencies responders get to their destination as quickly as possible.

I. Conclusion

In conclusion, we developed the IIS Java platform for the simulation of traffic flow on an isolated intersection. This platform helped us develop different STLS algorithms that help optimize traffic flow and minimize the delay time for road users. We started by introducing the real-world possible applications for our algorithms and presented prior researches on the subject. After, we presented our IIS platform, explained how our STLS algorithms function. Finally, we showcased their efficiency at managing different traffic flows. In this paper, we demonstrated that isolated intersections can be more efficiently managed by our STLS algorithms rather than those in previous researches and those used today.

Acknowledgment

This research was financially supported by the "Fonds Québécois de la recherche sur la nature et les technologies (FRQNT)." We would like to thank Caroline Houle, teacher at Collège de Maisonneuve, for her valuable comments.

References

[1] "TomTom TrafficIndex:Toronto", TomTom.com, 2019, [Online], https://www.tomtom.com/en_gb/trafficindex/city/toronto, [Accessed: 09/August/2019].

[2] M. Bsrth and K. Boriboonsomsin, "Traffic congestion and greenhouse gases", *Access Magazine*, 2009.

[3] IIS platform by M. Barri R. Gayta and J. Rezgui on GitLab, https://gitlab.com/reinergayta/26lumieresintelligentes [last visit and update 09/August/2019: open source code].

[4] A. Kanungo, A. Sharma and C. Singla, "Smart Traffic Lights Switching and Traffic Density Calculation using Video Processing", Recent Advances in Engineering and Computational Sciences (RAECS), 2014.

[5] National Association of City Transportation Officials, "Urban Street Design Guide: Signal Cycle Lengths", https://nacto.org/publication/urban-street-design-guide/intersection-design-elements/traffic-signals/signal-cycle-lengths/, 2013. [Accessed: 09/August/2019].

[6] M. Younes and A. Boukerche, "An Intelligent Traffic Light Scheduling Algorithm Through VANETs", *10th IEEE International Workshop on Performance and Management of Wireless and Mobile Networks*, 2014.

[7] J. Redmon , "YOLOv3: An Incremental Improvement", *University of Washington*, 2018

[8] Y. Zhou and al., "Image-based Vehicle Analysis using Deep Neural Network: A Systematic Study", *IEEE International Conference on Digital Signal Processing (DSP)*, 2016.

[9] I. A. Abbasi and A. S. Khan, "A Review of Vehicle to Vehicle Communication Protocols for VANETs in the Urban Environment", *future internet*, 2017.

[10] S.T. Le, "Feux de circulation – comptage des véhicules et des piétons aux intersections munies de feux", *http://donnees.ville.montreal.qc.ca/dataset/comptage-vehicules-pietons,* 2019. [Accessed: 09/August/2019].

[11] "Car dimesions of all makes with size comparison tool", https://www.automobiledimension.com/, 2019. [Accessed: 09/August/2019].