A Dynamic Traffic Scheduling System

Chenkai Shao, Suyang Hu, Siyuan Zheng

***Abstract*—The standard traffic light system uses a fixed schedule that switches between states of different colors with fixed time. While simple and predictable, this fixed schedule does not fully utilize the crossroad and often impose unnecessary wait time on vehicles and pedestrians. In this paper, we introduce a new traffic light scheduling scheme that dynamically schedules all the traffic lights at a crossroad based on the amount of traffic in all directions. Our scheduling scheme aims to minimize the average wait time of both vehicles and pedestrians while ensuring fairness between them at the same time. We design the scheme to be systematic and comprehensive so that it can be configured and adapted for different types of crossroads. As we are in the initial design stage of the new scheme, evaluations are performed in simulation software with real-life requirements and issues taken into account. Our experiments show that, for different traffic situations, our dynamic traffic light scheduling scheme consistently achieves significant improvement on the average wait time of traffic and the utilization of a crossroad. Fairness is evaluated and improvement is also seen in the experiment results.**

Keywords—traffic lights, scheduling algorithm, SUMO, waiting time

I. INTRODUCTION

Currently, most of the crossroads use traffic lights with fixed order and fixed time. Under this scheduling approach, it is common to see that a fair amount of vehicles are waiting for the green light in one direction while no vehicles are crossing or approaching the crossroad in the intersecting direction. In this situation, it is clear that the vehicles could have been able to pass the crossroad earlier without any danger if the traffic lights were more intelligent. While being an extreme example, it represents many more common situations where traffic is unnecessarily blocked simply because the traffic lights are static and not able to react to the traffic flow. With all the vehicles and pedestrians combined, a significant amount of people's invaluable time can be wasted. Therefore, we are in urgent need of a new traffic light scheduling scheme which can maximize the utilization of a crossroad so as to minimize the average wait time of vehicles and pedestrians.

It is true that we are seeing more and more technologies used on crossroads to make them smarter. Some car detection devices can be equipped on a lane and are able to activate the traffic light only when cars are detected. These technologies certainly improve the efficiency of crossroads but their usage is often sporadic and limited to a few lanes. Therefore we design a dynamic traffic light scheduling scheme that takes into account the traffic, including vehicles and pedestrians, from all the lanes at a crossroad and dynamically controls the traffic lights accordingly in real time.

A. Dynamic Traffic Light Scheduling

Traditional traffic lights are fixed in two aspects, 1) the order of states and 2) the length of each state. Some obvious improvements can be done by dynamically adjusting the length of each state. For example, a green light can be transitioned to red earlier so as to allow traffic from other directions if no traffic is approaching the crossroad on that lane. However, we go a step further by completely removing the order of the states. The states of traffic lights are no longer dependent on the previous state and, rather, they are dynamically generated by our scheduler. Based on the traffic conditions, vehicles and pedestrians on a set of non-intersecting lanes or crossings are allowed to pass the crossroad at the same time. The set of lanes and crossings are dynamically generated over time in a way that minimizes the overall average wait time. Therefore, neither the order of states nor the length of each state is static. By breaking the fundamental limitation of traditional traffic lights, we are able to maximize the utilization of a crossroad and save people's time. In the current stage, our scheme consists of abstracted and simulated hardware and a scheduling algorithm implemented.

B. Hardware Requirements and Abstraction

For our scheme, hardware is primarily required in two areas, 1) vehicle and pedestrian detection and 2) an embedded control system.

In order to control all the traffic lights at an intersection and makes global scheduling decisions, our scheme needs to monitor the traffic from all directions. For vehicles, we envision that cameras need to be installed on the traffic light poles, at least one for each direction. The cameras at the same crossroad need to be connected since they will communicate the detection of traffic in real time. Furthermore, we make the assumption that the cameras are installed with the specific configuration of this specific crossroad. The position of the cameras needs to be fixed and the coordinates of key features in the image that they capture, such as the bounds and width of a lane, need to be pre-calculated. Other properties of the crossroad, including the number and direction of the lane, are also assumed to be known to the camera. We may assume reasonable traffic detection results with modern computer vision technologies in that 1) road configurations do not change frequently, 2) once a camera is installed at a crossroad, it does not need to move, and 3) Software tools can be provided to ease the calculation of the coordinates. For pedestrians, we consider them as groups due to the difficulty in detecting them in a per person scale with a camera. In addition, pedestrians may be blocked by structures

such as traffic light poles and greens at the corner of a crossroad so we need a more reliable way of getting the information of pedestrians. For our simulation, we assume that the traffic light poles are equipped with a button so that pedestrians can press the button to indicate their presence. This is also a reasonable requirement since a good amount of crossroads already have such a button in the U.S. The algorithm will only try to optimize the waiting time of the pedestrians if they indicate their willingness of crossing by pressing the button, which is in line with the current purpose of the button at crossroads.

For the overall system, a relatively powerful embedded system needs to be used to process the video stream, invoke the scheduling algorithm and control the traffic lights. Due to the real-time nature obtaining traffic conditions and actuating traffic lights, a real-time operating system needs to be used to guarantee that the traffic lights are controlled on time. In addition, since high level functionalities such as computer vision is required, an embedded Linux system may be necessary. Combining all the requirements, we imagine a real-time version of embedded Linux operating system would be used to fulfill the tasks.

C. *Related Work*

In [1], the authors develop an algorithm based on the minimum destination distance first algorithm and minimum average destination distance first algorithm, which calculates the traffic of different roads and schedules the vehicles through the analysis of their destination. However, this algorithm proposed in [2] is not practical in real life, because it is often difficult to obtain information about the vehicle's destination, and these destinations may change in real time. Moreover, the lanes in real life are assigned to the directions at intersections instead of the distance of the vehicle's destination. Although it is hard to use this algorithm in current traffic situation, we can take it into consideration when dealing with autonomous vehicles in the future, at which time this algorithm would be more efficient and useful.

In [3], a real-time control algorithm is designed for traffic lights based on CPU process scheduling. Its important contribution is that it calculates the time a vehicle takes to cross an intersection, which ensures the precision of its analysis for real life scheduling problems. However, the algorithm proposed in [3] does not take into account the pedestrian crossing. Like many other scheduling approaches, the pedestrians waiting at crossroads are often ignored, which makes their average waiting time to be much longer.

Therefore, considering what the existing works fail to consider or accomplish, our traffic light scheduler takes into account both vehicles and pedestrians and base the design heavily on real life conditions and situations.

D. *Organization*

The rest of the paper is organized as follows. Section II illustrates the simulation environment and the model of a crossroad that we use for algorithm presentation and simulation. Section III discusses our dynamic traffic light scheduling algorithm in detail. In Section IV, we explain the design of the experiments and show the results. The limitations of our approach and future work are also discussed in Section V. Finally, the paper is concluded in Section VI.

II. Simulation and Model

We choose to use SUMO as our traffic simulation software for the reason that it provides more comprehensive functions and complete documentation than any other open source platforms [3]. It also ships with useful tools that can be used to generate maps and random traffic. We use the map generation tools to create a crossroad which has traffic from four directions, each containing three lanes (right turn, straight and left turn) for vehicles and one sidewalk for pedestrians. Thus, there are 12 possible routes for vehicles (3 for each direction) and 4 crossing for pedestrians, totaling 16 routes that may be controlled by traffic lights in Figure 2. Note that pedestrians from both directions on a single crossing is controlled by the same traffic light. This type of crossroad is illustrated in Figure 1 and we will call it the “standard” crossroad in the rest of the paper. We will show how our scheduling algorithm can be adapted to different types of crossroads and why our evaluation, though conducted on this “standard” crossroad, can be valid for all types of crossroads in Section III and Section IV.

The blue area on the lanes in Figure 1 represents the area of detection of the cameras that would be installed. In SUMO, they function as detectors that provides vehicle information during simulation. For the pedestrians, walking areas, which connects sidewalks to the crossings, and the crossings themselves provides the information of pedestrians.

In SUMO, the default traffic light simulation uses the tradition static scheduling. However, python scripts can be developed to interact with and control the simulation at runtime through a feature called TraCI. It provides APIs for the python scripts to step through simulation progress, obtain all kinds of simulation status and change the states of traffic lights. In SUMO, each crossroad has one traffic light element, which represents the traffic lights for all the lanes. The state of this traffic light elements consists of the states of all the traffic lights that it represents.

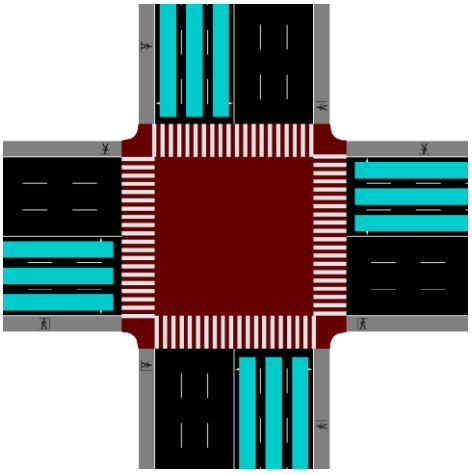


Figure 1. Standard Crossroad

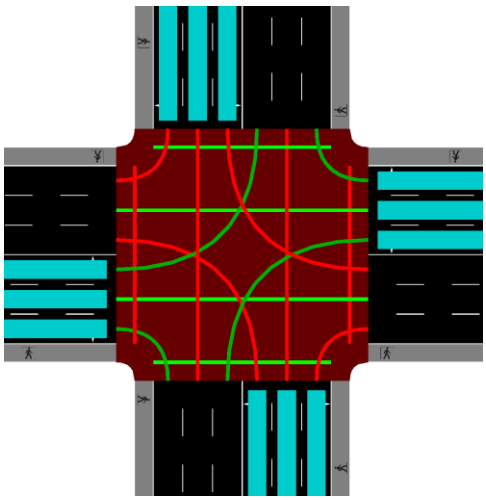


Figure 2. Routes Connection

In each simulation step, our python script obtains vehicle and pedestrian information, which in real life would come from the cameras and the push buttons, from the virtual detectors and walking areas. Our algorithm uses the information to generate a set of traffic light states that is optimal for current simulation step and sets the traffic light to this generated state. As long as the traffic lights are set correctly, SUMO automatically handles the acceleration and deceleration of traffic according to the traffic lights.

III. Scheduling Algorithm

Before giving a detail description of our algorithm, its main structure is shown in the following Figure 2.

To demonstrate our scheduling algorithm, we consider the "standard" crossroad described above. The standard crossroad has four directions, West, South, East and North. Each direction has 3 lanes, Left, Straight and Right as well as a crossing for pedestrians across the lanes. For vehicles and pedestrians at the crossroad, there are 16 possible routes controlled by 16 traffic lights. They can be represented by West - Left, West - Direct, West - Right, West - Pedestrian, South - Left, South - Direct, South - Right, South - Pedestrian, East - Left, East - Direct, East - Right, East - Pedestrian, North - Left, North - Direct, North - Right, and North - Pedestrian. Some of these routes are mutually exclusive with each other, which means they do not intersect each other. Thus, we can define a 2-D matrix with a size of 16 * 16 to represent the relations between the routes. For any value of i and j, if matrix[i][j] is 1, route i and route j are mutually exclusive; if matrix[i][j] is 0, route i and route j intersects; if matrix[i][j] is -1, i and j are the same. We can initialize this mutual exclusiveness matrix as follows.

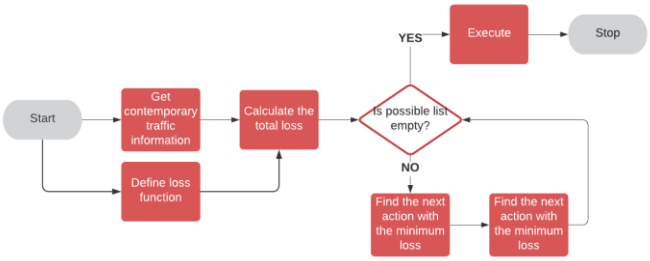


Figure 3. Algorithm Structure

$$
\begin{aligned}
A = &[-1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 0],\\
&[\ 1,-1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 1],\\
&[\ 1,\ 1,-1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1],\\
&[\ 0,\ 0,\ 0,-1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1],\\
&[\ 0,\ 0,\ 1,\ 0,-1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1],\\
&[\ 1,\ 0,\ 1,\ 1,\ 1,-1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0],\\
&[\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,-1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1],\\
&[\ 1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 0,-1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1],\\
&[\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 0,-1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1],\\
&[\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1,-1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1],\\
&[\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,-1,\ 0,\ 1,\ 1,\ 1,\ 0],\\
&[\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 0,-1,\ 0,\ 1,\ 1,\ 1],\\
&[\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 0,-1,\ 1,\ 1,\ 0],\\
&[\ 0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1,-1,\ 1,\ 0],\\
&[\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,-1,\ 0],\\
&[\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 0,-1]
\end{aligned}
$$

In order to find out the optimal traffic light states for the current time interval, we need not only the relations between the routes but also the contemporary traffic situation. In our algorithm, we use the time that vehicles and pedestrians have waited so far to represent the current state of traffic. For every moment, we assign a penalty value to each of the 16 routes based on how long there are vehicles or pedestrians waiting for the route in the detection area. To represent the penalty values, we define a 1-D matrix with 16 elements.

$$P = [P_0,\ P_1,\ P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}, P_{15}] \quad (1)$$

Let i be the index of the route out of the 16 routes, the corresponding element are set in the following way.

$$P_n = \begin{cases} 0, \ no\ vehicles\ or\ pedestrians\ on\ it \\ 1, \ the\ first\ time\ vehicles\ or\ pedestrians\ on\ it\ appers, \\ \quad or\ the\ vehicles\ or\ pedestrains\ on\ it\ are\ moving \\ (P_{n-1}, +1) * P_{n-1}, \quad otherwise \end{cases} \quad (2)$$

In order to ensure fairness between entities and eliminate the possibility of deadlocks, we design the penalty value to grow quadratically instead of linearly. Assuming that the value in the previous step is P_{n-1}, the value in the current step P_n should be set to $(P_{n-1}+ 1)* P_{n-1}$. We use this non-linear growth pattern to make sure the penalty value increase dramatically as time goes by so that no one would be blocked for too long. When traffic is allowed on a route in the previous step, the penalty is fixed to a low value.

In order to find out the optimal states of traffic lights for the current time period, our algorithm considers the cost and reward of a route if traffic is allowed on that specific route. The cost and reward values can be calculated from the penalty value introduced above. The reward value R for a route represents the benefit of allowing traffic on the corresponding route. It is defined as the same value of the penalty of the route.

$$R_i = P_i \tag{3}$$

To find out the first route in the optimal set of mutually exclusive routes, the cost value represents the sum of the penalty values of the routes that are blocked by the route we want to enable. To calculate this value, we need to modify the relation matrix A to Exclusive matrix so that E[i][j] is set to 1 only if route i and route j are exclusive. It can be initialized as follows.

E=[0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1],
[0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0],
[0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
[1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1],
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0],
[0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0],
[1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
[0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0],
[1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1],
[1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1],
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
[1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0]

From the matrix E above, we can calculate the cost value as such.

$$C_i = sum(E_i * P) \tag{4}$$

Therefore, the total loss value for a route is the difference between the reward and the cost.

$$L_i = C_i - R_i \tag{5}$$

For each simulation step, we iteratively find a set of routes with minimum loss value that are mutually exclusive. To break ties between routes that have the same loss value, we define a scheduling priority value to deal with the situation. The longer a route is disabled since it was enabled last time, the larger priority value it gets. The initial priority value for each route is assigned to 0. In every step, it increases by 1. However, when a route is enabled, the priority value is reset to 0 again. After that, we choose the one with a higher priority to break ties between routes with the same loss value. If there is more than one route with the same priority and loss, we perform routes in their index order by default.

For all the remaining routes in the set after we find the first one, we compute a different cost value, which uses another matrix S. Matrix S is similar to matrix A, but the element is 1 only if two routes are mutually exclusive and 0 otherwise. A matrix S derived from matrix E above looks like the following.

S = [0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0],
[1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1],
[1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1],
[0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1],
[1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0],
[1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1],
[1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1],
[1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1],
[0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0],
[1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1],
[0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0],
[0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0],
[1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],
[0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0]

Suppose we choose i as the first route, we define an 1-D array $F = S[i]$ that represents the set of routes that are still available to be considered. We then find the next route j to be enabled within only the routes whose value has been set to 1 in F, which means i and j are mutually exclusive. When we calculate the cost for route j, we exclude the routes that have already been blocked by route i. Therefore, we can calculate the cost for route j as such.

$$C_j = sum(E_j * F * P) \tag{6}$$

And the total loss is still the difference of Reward value and Cost value.

$$L_j = C_j - R_j \tag{7}$$

After finding the next route j with the least loss, we need to update the array F as F=S[i] * S[j]. Iterate this step until all the elements in F are 0, which means we have picked all the feasible routes for the current simulation step.

IV. Experiment Evaluations

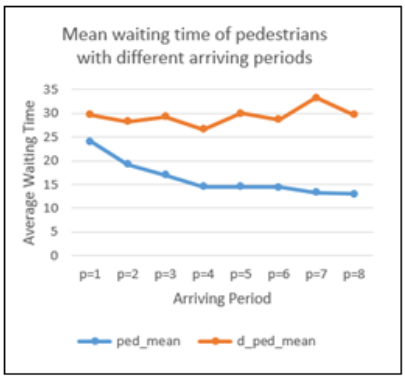
To evaluate the efficiency of our proposed scheduling algorithm, we conduct several comparative experiments to assess it in different aspects. In these experiments, we will test on various traffic configurations, and verify the improvement of our algorithm upon the default fixed time scheduling method. Our research mainly evaluate these two methods through the average waiting time of vehicles and pedestrians, as well as the waiting time deviation, so as to prove the efficiency and fairness of our scheduling algorithm. The experiments will be divided into four parts, which are shown as following:

A. Arriving Rate

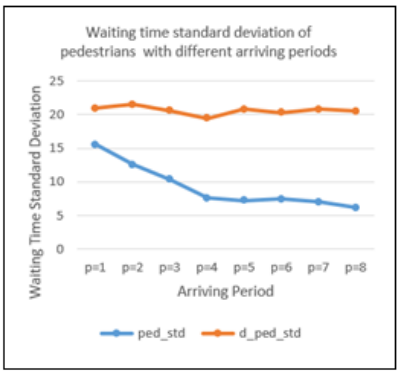
In this experiment, we test the performance of our scheduling method with different arriving rate. Here the arriving rate reflects the arriving period of vehicles and pedestrians, which indicates the arriving time difference between two objects. We adjust the time period from 1 second to 8 seconds, and the waiting time results of pedestrians are shown in the following Figure 4.

In Figure 4, the blue lines show the results of our proposed algorithm, and the yellow ones show the results of default fixed

scheduling time method. It is evident that the average waiting time of our method is shorter than that of the default method. Also, with the increase of object arriving periods, the average waiting time of pedestrians reduces significantly with our algorithm. However, since the default method adopts a fixed scheduling strategy, its average time is not affected obviously through changing the arriving rate of objects. In addition, the standard deviations of our scheduling algorithm are short and are much lower than the default ones, which could also show the fairness of our method.



(a) Average Waiting Time



(b) Waiting Time Standard Deviation

Figure 4. Different Arriving Rate

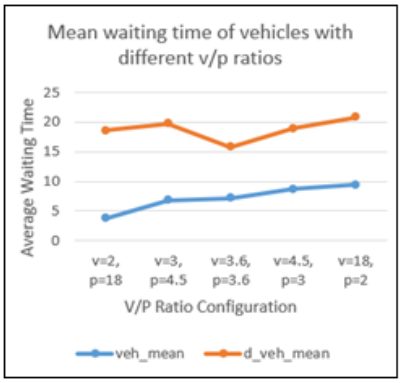
B. *Pedestrians/Vehicles Ratio*

This part of our research considers different pedestrians and vehicles configurations, in which we change the P/V ratios to test the adaptation of our algorithm in various traffic situations. For that in some crossroads there may be more vehicles, whereas in other areas such as school, pedestrians occupy a very high ratio. The experiment results are shown in Figure 5.

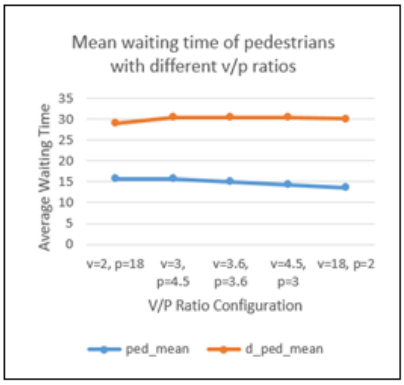
Here we set the scheduling time as 180s, and the total number of objects as 100. We can see that for different configurations of vehicle/pedestrian ratio, the average waiting times of both vehicles and pedestrians of our algorithm are significantly lower than those of the fixed time scheduling method. Therefore, our algorithm is suitable for various traffic situations, and this kind of change does not affect its performance in a high degree, which also proves the stability of our algorithm.

C. *Scheduling Length*

The third experiment evaluates our algorithm with different scheduling lengths, which means the total time that objects will be generated. To be more specific, here e=150 represents that the vehicles and pedestrians in the crossroads will keep arriving for 150 seconds. Figure 6 shows the performances results of both scheduling strategies.



(a) Waiting Time for Vehicles



(b) Waiting Time for Pedestrians

Figure 5. Different P/V Ratio

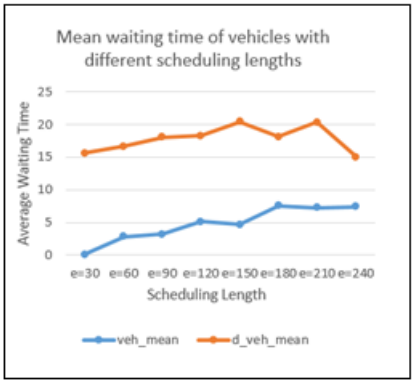
From Figure 6 we can see that, although the average waiting times of vehicles and pedestrians of our proposed algorithm increase a little when increase the total scheduling time, they are still significantly lower than those of the default fixed scheduling strategy. Our algorithm reduces the average waiting time to about 50% of the default method, and it shows good adaptability with different scheduling length as well as improving the utility efficiency of the traffic light systems.

D. *Some Common Mistakes*

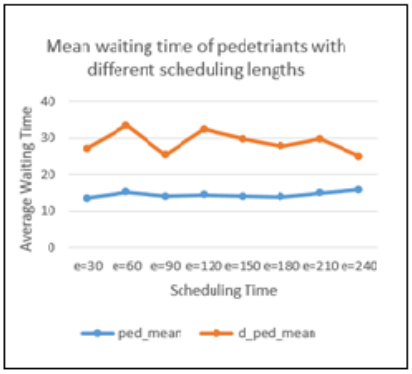
In this experiment, we evaluate our scheduling algorithm through applying different object arriving densities. Here the density refers to the maximum number of objects that could arrive in the crossroad at a certain time point, which is different from the arriving rate indicator used in experiment A. We fix other parameters and adjust the object density from 2 to 8, and the comparative experiments are shown in figure 7.

Through calculating the average waiting time of both vehicles and pedestrians, it is obvious that these results are consistent with those of above experiments. The performance of

our algorithm are not affected much by changing the object density, and the waiting time of vehicles and pedestrians are lower than the fixed scheduling method in a very high degree. Thus, the stability and outperformance of our algorithm could be proved.



(a) Waiting Time for Vehicles



(b) Waiting time for Pedestrians

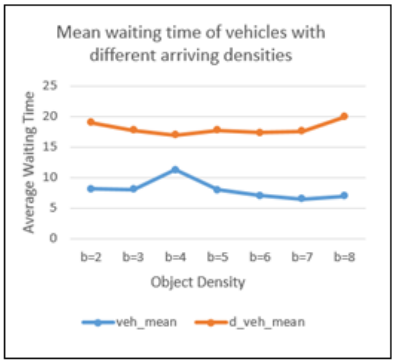
Figure 6. Different Scheduling Length

V. Limitations and Future Work

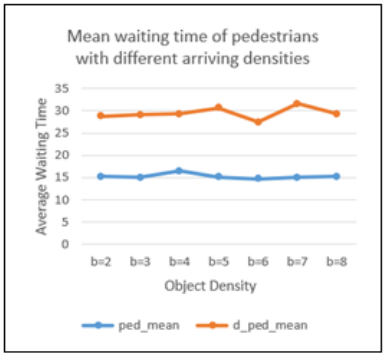
Although we largely base our assumptions on real life situations, there still exists inevitable discrepancies between our simulation and real life. Currently, we do not have yellow lights in our simulation, which means the traffic lights switch between red and green immediately. This causes vehicles to decelerate and stop in the middle of the crossroad when them do not have enough time to react to the change of the traffic lights. Future work will need to be done in the algorithm and simulation model to incorporate yellow lights.

Furthermore, we are assuming perfect vehicle detection results from the cameras in our simulation. However, object detection based on computer vision techniques suffer from errors and the limitation on resources available on embedded real time systems leads to more errors. We will need to research the state-of-the-art computer vision technologies that are suitable for the capability of potential hardware platform that may be used. With these in mind, we need to find out the rate and characteristics of errors in detection and inject the errors in the same way in our simulations. Although the detectors in SUMO are by nature perfect, we could generate impose errors onto the perfect information obtained from the SUMO detectors in the python scripts. Afterwards, we would experiment and evaluate the impact of the errors on the scheduling efficiency.

Another remaining work is utilizing traffic information from the neighbor crossroads. In order for the scheduler at one crossroad to acquire information from adjacent crossroads, cameras will need to be connected through Ethernet or WIFI so that the traffic information can be transferred between crossroads. Communication latency will need to be considered for the real time requirement. With information from adjacent crossroads, our scheduler will be able to predict the traffic flow in all directions. As a result, the impact of errors from computer vision traffic detection will be mitigated and the efficiency can be further improved.



(a) Waiting Time for Vehicles



(b) Waiting time for Pedestrians

Figure 7. Different Object Densities

VI. Conclusion

In conclusion, this paper solve the traffic lights scheduling problem to minimize the waiting time of vehicles and pedestrians as well as ensuring fairness between them. Firstly, we develop a novel scheduling algorithm, in which we figure out the mutual relationships between different actions, and then defined the cost and reward for each action to determine which action will be executed at next a time step. Secondly, we combine this algorithm with SUMO platform, in which we build the standard crossroads and use some tools to generate cars and pedestrians. Finally, we conduct some comparative experiments to evaluate the performance of our algorithm in different environment configurations. We consider indicators such as

arriving rate, arriving density, scheduling length, and ratio between vehicles and pedestrians. Through analyzing these results, we can safely conclude that our algorithm performs well in different situations, and it outperforms the fixed time scheduling method.

REFERENCES

[1] F. Ahmad, S. A. Mahmud, G. M. Khan and F. Z. Yousaf, "Shortest remaining processing time based schedulers for reduction of traffic congestion," *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, Las Vegas, NV, 2013, pp. 271-276. doi: 10.1109/ICCVE.2013.6799805.

[2] Jinrong Zhou, Xiaofang Zhou, Zhibin Chen and Xiao Chen, "Research of real-time control algorithm for traffic lights based on CPU process scheduling," *2011 IEEE International Conference on Anti-Counterfeiting, Security and Identification*, Xiamen, 2011, pp. 110-114. doi: 10.1109/ASID.2011.5967428

[3] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. *Recent Development and Applications of SUMO - Simulation of Urban MObility*. International Journal On Advances in Systems and Measurements, 5 (3&4):128-138, December 2012.