IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 3, 2018, accepted September 7, 2018, date of publication September 26, 2018, date of current version October 17, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2871337

Autonomous Intersection Management: A Heuristic Approach

AADITYA PRAKASH CHOUHAN, (Student Member, IEEE), AND GOURINATH BANDA, (Member, IEEE)

Discipline of Computer Science and Engineering, IIT Indore, Indore 453552, India

Corresponding author: Aaditya Prakash Chouhan (phd1501201006@iiti.ac.in)

ABSTRACT Self-driving cars do not sound like a part of science fiction movies anymore. With most of the automobile giants committed to launching their autonomous cars as soon as possible, it is expected that self-driving cars will hit the road in a very short time. Increasing the penetration factor of autonomous vehicles will need algorithms to control them in an efficient way in different scenarios. One such scenario is intersection management (IM). We propose an intuitive heuristic to resolve space–time conflicts in vehicle's trip, enabling vehicles to cross the intersection safely, and with minimum delay. The intersection is modeled as a group of conflict points where intersecting internal links cross each other. For safe and efficient scheduling of vehicles, heuristics are proposed separately for: 1) entering in approach lane; 2) safe traversal in the approach lane; 3) safe and efficient crossing of intersection; and (4) safe departure along depart lane. Vehicle scheduling is done using a three-leveled heuristic in which each level serves a distinct purpose and together guarantee collision-free passing of vehicles. Performance evaluation of the proposed scheme is done in Simulation of Urban MObility simulator and a comparative analysis is done with three other IM schemes. Experiments show that the proposed scheme gives a smaller average trip delay of vehicles compared with other IM schemes.

INDEX TERMS Autonomous vehicles, collision avoidance, heuristic algorithm, intersection management.

I. INTRODUCTION

Vehicular technology is undergoing a major trans- formation at this time and technological advancement in sensors, wireless communication, control strategies and data management strategies are the drivers of this transformation. Upcoming vehicles shall be equipped with the capability to communicate with surrounding vehicles and infrastructure (typically consisting of control units attached on the roadside or Road Side Units (RSU)) which enables them to follow the optimum driving strategy that is either decided by mutual negotiations of vehicles or by a central controller. Both the ways, however, will follow some protocol that ensures that scheduling of vehicles is done in the most efficient and safest manner irrespective of the driving scenario.

Intersection management is one of the crucial areas of traffic management. Intersections are also known as 'The Bottlenecks of Traffic'. Manually driven cars result in low efficiency at intersections because humans have a large reaction time and lack of cooperation. Autonomous cars, on the other hand, are way more cooperative and have a fast reaction time. Consider the case of an intersection where all the vehicles in a queue can stop or accelerate at the same time; this way, number of vehicles passing in one phase will be maximum. This can happen when all the vehicles are autonomous and can accelerate or decelerate simultaneously.

Full autonomy of vehicles will be a reality in future as the advantages offered by autonomous cars are huge. They will result in a steep decrease in human casualties due to traffic, efficient traffic management hence less pollution, less congestion, less economic loss and less human time loss. These advantages are so significant that various government organizations across the globe have accepted autonomous vehicles as a solution to most of the current traffic management problems and are also creating awareness among people about the same. Organizations such as Department of Transportation (DoT) [1], American Association of Motor Vehicle Administrators (AAMVA) [2] etc. have set guidelines for autonomous vehicles regarding safety and performance. Also, the Society of Automotive Engineers (SAE) has defined various levels of autonomy a car can achieve [3]. It contains six different classes ranging from 0 to 5 with the level of autonomy increasing with the number.

We present an IM algorithm that considers a traffic consisting only of autonomous cars. In this algorithm, vehicle's

arrival at the intersection is scheduled by varying its velocity. The velocity given by the algorithm is fixed and the vehicle travels throughout the intersection area with that velocity only excepting the initial region, where it has to perform the transition from initial velocity to the assigned velocity. This velocity is obtained after resolving all the conflicts that the vehicle might have in the intersection. Since velocity fluctuation is known to cause a disturbance in traffic which sometimes leads to jams and even accidents, having a constant velocity traversal also adds stability to our system.

IM is a resource sharing problem where the space of the intersection is the resource being requested by vehicle and the task is to make space-time reservations depending on the expected arrival times of different vehicles such that no two vehicles occupy same space at the same time. In other words, the task is to resolve space-time conflicts of vehicles to prevent collisions such that average delay caused by this scheduling is minimum. As proved in [4], scheduling vehicles incoming to an intersection is an NP-Hard problem, it motivates us to present a heuristic algorithm for intersection management. This algorithm being a heuristic one, has very less computational requirements, making it suitable for real-time applications. The aim of this paper is to propose an intuitive approach to intersection management that is based on a heuristic and does not use any computationally intensive optimization procedure.

Rest of the paper is organized as follows. At first, in Section II, we present a brief review of literature relevant to autonomous intersection management. In section III, we describe the architecture used, infrastructural requirements and assumptions made in the proposed scheme. Intersection model is also described in this section. In section IV, we present the heuristic that schedules vehicles in the intersection. In section V, we present the simulation results and outcome of comparison with three other schemes. In section VI, we draw conclusions from the results and also discuss about future direction of work.

II. RELATED WORK

Vehicular intersection management has a rich literature dedicated to improving it's efficiency in scheduling vehicles and overcoming conflicts. This section contains overview of some selected works in this regard. Presented works are classified into three categories, which are: (i) Traffic light optimization (both online and offline); (ii) Intersection management of connected vehicles, and (iii) Intersection management of autonomous vehicles.

A. TRAFFIC LIGHT OPTIMIZATION

Traffic lights control most of the controlled intersections globally. Performance of traffic light can be improved by optimizing the phase cycle durations depending on the average traffic on different constituent phases. This can be done offline as well as online. Nieto *et al.* [5] present an offline traffic light cycle optimization using Particle Swarm Optimization. When traffic lights have fixed cycle durations, vehicles have to wait for green signal even if there is no other conflicting vehicle present. When the traffic light cycle can be adapted based on the current traffic conditions, it is known as Adaptive or Dynamic traffic light. Reference [6]–[9] present works towards dynamic traffic lights.

B. INTERSECTION MANAGEMENT OF CONNECTED VEHICLES

With the introduction of Vehicular communication networks, vehicles become capable of sharing information with other vehicles and infrastructure. This information can be used by traffic management services. Azimi *et al.*[10]–[12] present how this communication can be used for transportation system as a whole and to intersection and roundabout management as well. In a situation where every vehicle is equipped with a communicating device, we might not need a traffic light at all; rather vehicles can talk to themselves and sort their travel accordingly. This idea is proposed in [13], in which, vehicles elect one of the vehicles as the leader. The leader will be responsible for scheduling traffic through that intersection thus acting as 'The Virtual Traffic Light.'

For any intersection management algorithm, scheduling vehicles safely is the most important requirement and the algorithm must satisfy the no-collision condition. Various dedicated works have been established for avoiding collision of vehicles at intersection under a connected vehicle environment as in [17], [18], and [26].

C. INTERSECTION MANAGEMENT OF AUTONOMOUS VEHICLES

With introduction of autonomous vehicles, new possibilities towards traffic management opens and it is evident by a significant amount of publications dedicated to traffic management of completely autonomous traffic. One of the aspects of autonomous traffic management is intersection management. Some of the bench mark works done in this regard are at University of Texas at Austin. Dresner and Stone [23] approached the intersection management problem with a multi-agent approach. They proposed a First Come First Serve (FCFS) policy of vehicle scheduling in their work. Intersection area was modeled as consisting of square blocks and the vehicles need to call ahead for reservation of the blocks depending upon their arrival time and trajectory of travel. The number of divisions of the intersection area into blocks depends on the granularity. The resolution and the complexity of this algorithm increases with granularity. Further improvement is done in [24] by allowing vehicles to accelerate in the intersection square. In another work [25], an auction based intersection management scheme is proposed in which vehicles approaching an intersection can bid for fast passage on behalf of the passenger. To prevent this scheme from becoming biased towards wealthy driver agents a benevolent system agent is deployed to regulate these auctions.

Zhang *et al.* [19] propose a state driven priority based scheduling of vehicles at the intersection. They introduced a

new priority scheduling algorithm known as *sPriorFIFO* to reduce delay in high priority vehicles because of low priority vehicles ahead of them.

Lee and Park [27] solve intersection management problem using an optimization approach by formulating it as a constrained nonlinear optimization problem. Zohdy *et al.* [28] present a new tool for optimization of autonomous vehicle movements at the intersections known as Intersection Management using Cooperative Adaptive Cruise Control or iCACC. In their work, intersection is modeled as a group of possible conflicting points where vehicles can have conflict and their arrival at those conflicting points are scheduled by an optimization module that minimizes the total intersection delay of vehicles.

Parker and Nitschke [29] present an intersection management scheme with a decentralized neuro-evolution approach where vehicle controllers adapt to collectively navigate through the intersection. Their work uses Neuro-Evolution (NE) to automate the synthesis of collective driving behavior.

Miculescu and Karaman [30] present a polling based coordination policy for autonomous intersection control. More of such work dedicated to intersection management of autonomous vehicles can be found in [14]–[16], [21], and [22].

We now propose a heuristic algorithm for intersection management of autonomous vehicles. Heuristic algorithms in general, have very little computational requirement as they does not involve any processing intensive procedure. This makes heuristic algorithms better candidate for real-time applications than optimization schemes if other performance requirements are also fulfilled. Furthermore, the intersection model used in the presented work contains only four conflict points where scheduling is to be done, which is a smaller number as compared with other IM schemes. This later results in a smaller scheduling delay.

III. ARCHITECTURE

A. INTERSECTION MODEL

The intersection model used, consists of four roads converging to form a four-road intersection. Each road contains bidirectional traffic flow with 3 lanes in each direction. For simplicity, we consider roads to be straight and the intersection to be square. All the four roads approaching towards intersection have a buffer area at the entry. Vehicles entering the intersection are scheduled by a Central Vehicle Scheduler (CVS) which is responsible for resolving conflicts at the intersection. CVS resolves conflict by adjusting vehicle's velocity such that no two vehicles occupy same area at the same time. After calculating the velocity with which the vehicle has to travel all the way after the buffer region, CVS sends it to the concerned vehicle. As vehicles are required to start traversing with the assigned velocity as soon as they leave buffer region, CVS has to solve for the best conflict free velocity (V_{final}) for the vehicle by the time vehicle is in the buffer region. Following the buffer, is an *approach area* that leads to the intersection. The dimensions of: buffer, approach, lane and intersection area are as shown in Figure 1.

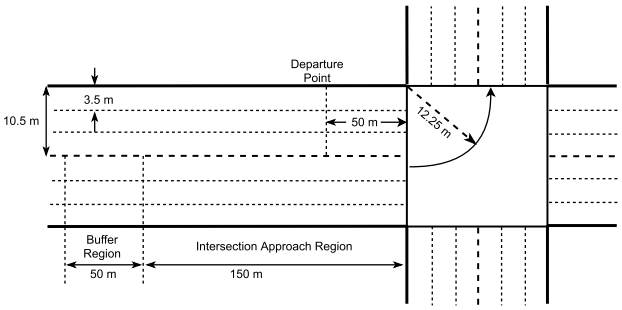

FIGURE 1. Figure containing infrastructural constants.

In the intersection area, vehicles have to fulfill some driving restrictions. These restrictions result in a better organization of vehicles. Firstly, vehicles have to travel in the lane corresponding to their destination direction making exclusive left turning, right turning and straight going lanes. Making exclusive lanes also rules out overtaking of vehicles in the lane. That means vehicles in a lane exit the intersection in the order they entered. U-turns at intersections are not considered. Vehicles have to travel along a fixed trajectory inside the intersection area. This trajectory depends on the geometry of intersection and we have considered it to be quadrant of the circle connecting source and destination lanes for non-straight going lanes. Also, we assume that the traffic is already organized and vehicles are traveling on the lane corresponding to their destination.

In summary, assumptions made in this work stand as following:

i) All vehicles are autonomous and are equipped with a communication device;
ii) There is no transmission delay and packet loss. In other words, communication performance is assumed to be perfect;
iii) Vehicles travel only in the lane corresponding to their destination direction, also ruling out lane change in the intersection area;
iv) U-turns at the intersection are not considered;
v) Trajectory of vehicles turning at the intersection is fixed;
vi) Slips and slides of vehicle tires does not take place in the study and
vii) Passenger cars only. Motorcycles, bicycles and pedestrians are not considered.

With lane and trajectory restrictions in place, the 4-road or 12-lane intersection transforms into 12 lane to lane connections intersecting at some fixed points. These intersecting points are termed as Conflict Points (CP). In the presented scenario, there are a total of 16 CPs and each lane to lane connection has 4 CPs. Exclusive right lanes have no

conflict points. With the above assumptions, our intersection will look as shown in Figure 2.

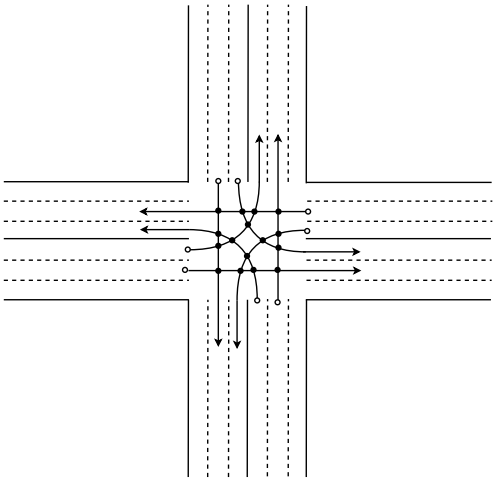

FIGURE 2. Intersection model.

With the simplified form of intersection, the task of CVS is to schedule vehicles at the 4 CPs in their route by varying their velocity such that no two vehicles are present at the same CP at the same time.

Table 1 gives representation given to different parameters in this work.

TABLE 1. Symbols used.

Variable symbol	Definition
$VehId$	Vehicle identification
$VehLen$	Vehicle length
$VehAcc$	Vehicle acceleration
V_{enter}	Entering velocity
V_{max}	Maximum allowed velocity
V_{lane}	Maximum safe lane velocity
V_{ref}	Window scheme reference velocity
V_{final}	Velocity returned by CVS
V_{temp}	Temporary iterating velocity variable
L_s	Source lane
L_d	Destination lane
cp_n	n^{th} conflict point

B. VEHICLE CHARACTERISTICS AND MODULES

Autonomous passing of vehicle through intersection involves autonomous control over it's longitudinal and lateral movement. Since these requirements are fulfilled by Level 2 or higher level autonomous vehicles, the presented algorithm targets vehicles with an autonomy of Level 2 or higher. For details about levels of autonomy, refer [3]. As soon as a vehicle arrives in the buffer, it has to send a message packet to the CVS containing following information.

$$\langle VehId, L_s, L_d, V_{enter}, VehLen, VehAcc \rangle$$

The environment consists of only autonomous vehicles entering in an intersection from 4 directions and leaving intersection along 4 directions. The Central Vehicle Scheduler (CVS) varies the velocity of vehicles to prevent any collision and does that in a very efficient way. To make this happen, vehicles have to send a message to CVS containing their journey details, along with entering velocity and time using Dedicated Short Range Communication (DSRC) protocol for wireless communication.

Along with vehicle and CVS as the two ends of the communication channel, in between, we also have Road Side Units (RSU) as a part of the communication module. These RSUs gather information from nearby vehicles and pass them to CVS and vice versa. RSUs also prompt vehicles regarding start and end of buffer region. In other words, RSUs tell vehicles to (1) pass message packet and (2) start velocity transition at the end of the buffer.

IV. RESOLVING CONFLICTS

To resolve conflicts, CVS extracts vehicle details from the received message packet and starts resolving conflicts in its path, if any. To detect conflicts, CVS performs internal simulations to calculate arrival time of the vehicle at the departure point and conflict points in it's path using the Newton's equations of motion. These arrival times are then compared with previous reservations, if there is a conflict, velocity is adjusted and the cycle is repeated. Lane and intersection conflicts are resolved using the heuristic presented in this section.

A. LANE VELOCITY CALCULATION

Conflict resolving for any vehicle starts with prevention of collision with the leading vehicle in the lane. The safe lane velocity (V_{lane}) of vehicle is only dependent on the leading vehicle's velocity profile. So, in the calculation of V_{lane} we only need to decide the vehicle's velocity which does not result in simultaneous occupancy of some intersection region with the leading vehicle. The strategy employed to prevent lane collision is to schedule the vehicle such that it always crosses the departure point after the leading vehicle. To do this, the algorithm is initialized with a velocity equal to the maximum allowed velocity (V_{max}) and iterated until we do not resolve lane conflict. The departure point is kept at a distance from the intersection square such that vehicles exiting the square with a velocity tending to zero will need to accelerate to the maximum allowed velocity. With the used maximum velocity and acceleration this distance comes out to be little less than 50 meters. So, we take 50 meter mark as the departure point. Algorithm to calculate V_{lane} is shown below in Algorithm 1. The *find_exit_time*() function in algorithm performs internal simulation using Newton's equations of motion to calculate exit time with given velocity. *lastVehExitTime* gives stored value of the depart time

Algorithm 1 LANEVELOCITY Finds Maximum Safe Lane Velocity

Input: $I = \{VehId, L_s, L_d, V_{enter}, VehLength, VehAcc\}$
Output: $O = \{VehId, V_{lane}\}$

$V_{temp} \leftarrow V_{max}$
while $(find\_exit\_time(V_{temp}) \leq lastVehExitTime[L_s])$ **do**
  $V_{temp} \leftarrow V_{temp} - \Delta V_{temp}$
$V_{lane} \leftarrow V_{temp}$
return V_{lane}

of the previous vehicle in its lane. ΔV_{temp} is a fixed value of 0.01 m/sec.

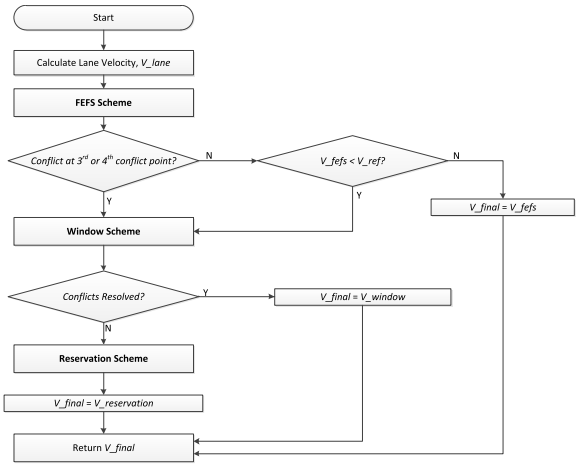

FIGURE 3. Decision Flow.

B. RESOLVING CONFLICTS AT INTERSECTION

To resolve conflicts at the intersection, V_{lane} is taken as the starting point for internal simulations and conflict detection. If there is no conflict, V_{lane} is set as the final velocity (V_{final}), else it goes through the adjustment pipeline. Pipeline is defined by the proposed heuristic and contains three steps. In the first step, we follow a *First Enter First Serve (FEFS) scheme*, in the second step we have a *Window scheme*, and in the third step we follow a conventional reservation scheme here termed as *Reservation scheme*. Second and third steps in this pipeline are applied conditionally. The conditions under which they are called will be specified in the respective sections. In a broad sense we can say that, FEFS scheme (first step), is not a completely collision free scheme when used alone, but returns V_{final} for most of the vehicles. Whereas, Window scheme (second step) is called when velocity assigned by FEFS scheme could be improved further or if FEFS scheme failed to give a collision free velocity and also to avoid piling of delays caused by FEFS scheme as explained in the respective section. The third and the last step is applied when both FEFS and Window scheme, return non-conflict-free velocities. So, we can say that the FEFS scheme's aim is to get the V_{final} with minimum computation, Window scheme's aim is to further reduce the average delay of vehicles and Reservation scheme's aim is to guarantee the no-conflict condition. Flow chart shown in Figure 3 explains the complete decision flow.

We next present these three steps in the order they are called.

1) FEFS SCHEME

First Enter First Serve scheme is the first scheme used to schedule vehicles at the Conflict Points (CP) in it's path. FEFS scheme assumes that at every CP, there is a single register-pair that holds arrival and departure time of next vehicle crossing it. For scheduling a new entered vehicle, it's expected arrival and departure time at every CP in it's path are calculated by internal simulation. These timings are then compared with timings in the register pair of CPs. This register pair is actually the top of a reservation stack that maintains record of recent reservations. New vehicles are always scheduled to arrive at corresponding CP after the reservation present in the register. However, comparison is not done for all the CPs in the vehicle's route, rather at only first two. A newly entered vehicle will adjust its velocity to arrive after the present reservations at the first two CPs in its path. If this velocity does not result in conflict at third and fourth CP, it is accepted as a safe velocity, otherwise, we call the Window scheme to check for a window in past reservations using the window scheme. Window scheme is also called in the case when the velocity returned by FEFS scheme is collision free but is less than a defined reference velocity value. This value could be any fixed velocity value but we define it as the average of final velocities (V_{final}) of all the vehicles scheduled so far in the simulation.

In simple words, FEFS scheme schedules vehicles by resolving conflicts at only first two conflict points. Also, this scheme only checks the top entry of the reservation stack and resolves conflict by scheduling the new vehicle to arrive after that last entry in the stack. As the vehicles are assigned reservations in the order they enter the intersection area, and FEFS scheme assigns new vehicles reservations after the previously entered vehicles, we name this scheme as First Enter First Serve. Algorithm for FEFS scheme is given in Algorithm 2. Functions *find_cp*1*_time*() and *find_cp*2*_time*() used in Algorithm 2 perform internal simulation using Newton's equations of motion to calculate arrival time at the respective conflict point using the given velocity. The register *cp_register* contains arrival time and departure time of the vehicle that crossed or is scheduled to cross that CP lastly. Departure time of a vehicle at any CP is its arrival time at that CP plus the passing time, which is equal to length of the vehicle divided by its velocity. This is how the length of the vehicle is incorporated.

Algorithm 2 FEFS: Finds Velocity for Vehicles After Resolving Conflicts at First and Second CPs

Input: $I = \{VehId, L_s, L_d, V_enter, VehLength, VehAcc, V_{lane}\}$
Output: $O = VehId, V_{FEFS}$

$V_{temp} \leftarrow V_{lane}$
$cp1\_clear \leftarrow 0$
while $(cp_1\_clear == 0)$ **do**
  $cp1\_time = find\_cp_1\_time(V_{lane})$
  while $(cp_1\_time < (cp_1\_register + safety\_gap))$ **do**
    $V_{temp} \leftarrow V_{temp} - \Delta V_{temp}$
    Update cp_1_time
  $cp_1\_clear = 1$
  $cp2\_time = find\_cp_2\_time(V_{lane})$
  while $(cp_2\_time < (cp_2\_register + safety\_gap))$ **do**
    $cp_1\_clear = 0$
    $V_{temp} \leftarrow V_{temp} - \Delta V_{temp}$
    Update cp_2_time
$V_{FEFS} = V_{temp}$
return V_{FEFS}

2) WINDOW SCHEME

The FEFS scheme intentionally does not check for collisions at the third and fourth CP. This makes FEFS fast but not a collision free scheme. A collision may occur in case when at the third or fourth CP a previously entered slow moving vehicle is having reservation and a new entered vehicle that have resolved conflicts at first two CP and is expected to arrive at the corresponding CP at the same time. There is one more issue with the FEFS scheme and it is about the piling of delays. *Piling delay* phenomenon occurs in the FEFS scheme because if a vehicle suffers delay from any reason, then, it may result in the delay being added in every vehicle's trip which is entering in close interval and sharing a CP with it and so on. This piling of delays may become very significant at high traffic density and some mechanism is needed to break this delay pile before it gets too large. This is done by the Window scheme.

Unlike the FEFS scheme, Window scheme makes use of a reservation stack and allows vehicles to pass the CP before previously entered vehicles if they can. Like FEFS scheme, Window scheme also takes V_{lane} as its starting point. In Window scheme, a window is searched between the recent reservations. If for a particular velocity, we get a window between reservations at every CP, we accept it as the collision free velocity.

There are two situations in which Window scheme is called. First when FEFS scheme could not generate a collision free velocity for the vehicle and second, when the FEFS scheme could generate a conflict free velocity but the resulting velocity is less than the set reference velocity here taken as the average of final velocity (V_{final}) of all the scheduled vehicles. This way in the first case, the Window scheme attempts to resolve conflicts unresolved by FEFS scheme by searching for a window in recent reservations for the vehicle to pass and in the second case attempts to find a greater velocity than that assigned by the FEFS scheme. This way, Window scheme breaks the delay pile as soon as it is generated. The algorithm for window scheme is shown in Algorithm 3. The *findWindow*() function in Algorithm 3 returns true when the corresponding CP is available if the vehicle travels with the passed velocity otherwise, it returns False. To obtain V_{window}, there must be a window at all the four conflict points.

Algorithm 3 WINDOWALGO

Input: $I = V_{lane}, V_{ref}$
Output: $O = V_{window}$

$Vel\_wo\_window = V_{FEFS}$ **if** $Velocity\_wo\_window < V_{ref}$ **then**
  $cp1\_clear, cp2\_clear, cp3\_clear, cp4\_clear \leftarrow 0$
  $V_{temp} \leftarrow V_{lane}$
  while $cp1\_clear\&cp2\_clear\&cp3\_clear\&cp4\_clear$ **do**
    $cp1\_clear = findWindow(cp1, V_{temp})$
    $cp2\_clear = findWindow(cp2, V_{temp})$
    $cp3\_clear = findWindow(cp3, V_{temp})$
    $cp4\_clear = findWindow(cp4, V_{temp})$
    $V_{temp} \leftarrow V_{temp} - \Delta V_{temp}$
$V_{window} \leftarrow V_{temp}$
return V_{window}

Figure 4 depicts working of window scheme. Four parallel bars represent reservation stack at four conflict points in a vehicle's path. Blue boxes (dark boxes) in these bars represent reservations in these stacks. For searching a window between reservations, window scheme starts with line 'A', which is corresponding to the initial velocity of window scheme i.e. V_{lane}. If a window is not available with that velocity, velocity is decreased and searched again. In this process, line A moves towards line B. As soon as a window is found at all the CPs as with line 'C', window search terminates and the corresponding velocity is returned.

3) RESERVATION SCHEME

Even with FEFS and Window schemes combined we do not have guaranteed collision free scheduling of vehicles because there may be a case in which Window scheme is not able to find a window for vehicle having conflict at third or fourth CP. For such cases we have this Reservation scheme, which is the traditional reservation policy. Reservation scheme starts with V_{FEFS} and adjusts velocity until at all the CPs conflicts are resolved.

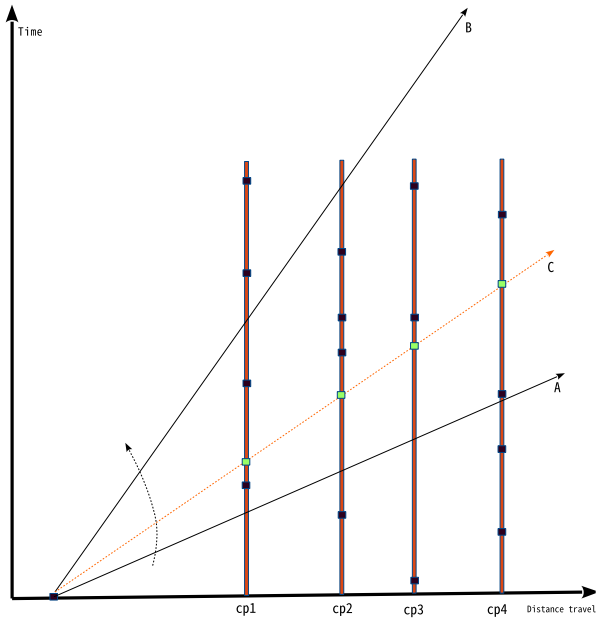

FIGURE 4. Window scheme.

C. DISCUSSION ON STEPS

The three steps described above work together to realize an efficient and collision free autonomous intersection manager. Every step has it's own purpose. They are discussed below.

1. FEFS scheme's purpose is to minimize the required computation for scheduling of vehicles. In the FEFS scheme to minimize the computation, only first two conflict points are checked for conflict. The reasoning behind this is that when the traffic density is at the lower side, delay in vehicle's trip is very less and most of the vehicles pass through with the maximum velocity allowed. This naturally makes vehicles entering first to leave first, added with time interval between arrival of two vehicles, because of less traffic makes conflict at third and fourth CP rare and to take care of these rare occurrences, we have next two steps. Hence, the FEFS scheme schedules most of the vehicles at low traffic level.
2. Window scheme's purpose is to further reduce the delay caused by the FEFS scheme along with attempting to resolve conflicts unresolved in FEFS scheme. Window scheme works independently of the FEFS scheme and vehicles scheduled by the window scheme do not interfere with FEFS scheme. Window scheme just looks for window in the reservation stack and assigns if it gets it. Window scheme become crucial at high traffic, because the delay pile phenomenon in FEFS scheme which is not visible at low traffic becomes significant at high traffic. Window scheme allows later entering vehicles to pass through the window and breaks the delay pile and prevents it from getting accumulated.
3. Reservation scheme's purpose is to guarantee that there is no collision at all. Even with FEFS and Window scheme combined, we may have some cases where conflicts are not resolved. Reservation scheme follows traditional reservation policy and assigns CP timing to the vehicle only when all the conflict points are available.

Vehicle will transit to the maximum allowed velocity as soon as it leaves the intersection square.

V. SIMULATION AND RESULTS

The described scenario is realized and simulated in SUMO traffic simulator. SUMO (Simulation of Urban MObility) is an open source, highly portable, microscopic road traffic simulation tool. In SUMO, a scenario is built by combining various input files containing specifications of nodes, edges, connections, routes and network [31]. All these files are combined in a configuration file to start the simulation. TraCI library is used to interact with the simulation and vehicle value setting and retrieval. A screen shot of simulation is given in Figure 5

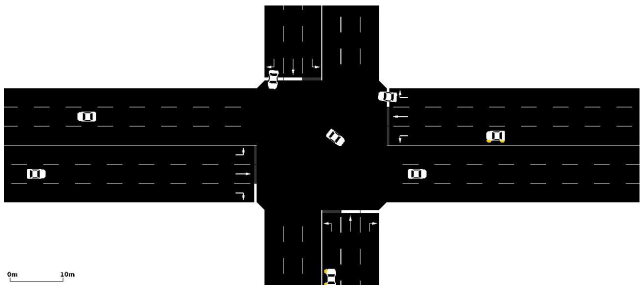

FIGURE 5. Simulation screenshot.

Along with the proposed heuristic, we have also implemented, on the same platform, two more autonomous intersection control algorithms. First one is the FCFS reservation policy as stated in [23] with intersection modeled as a group of square blocks and these blocks are reserved for incoming vehicles depending on their time-line. A granularity of 48 is taken, that means, intersection is modeled as a group of 48×48 square blocks.

The second scheme implemented is the CIVIC algorithm [27]. CIVIC algorithm solves intersection management problem using nonlinear constrained optimization by minimizing the length of overlap of trajectories of vehicles inside the intersection area. Length of overlapping trajectories is calculated by using Newton's equations of motion and Objective function is calculated by adding these overlaps for each pair of vehicles on conflicting lanes. Also, the phase conflict map in [27] is modified according to the phase numbers used in the intersection model.

These two algorithms belong to two different approaches towards autonomous intersection management namely multi-agent approach and nonlinear constrained optimization approach respectively, and that is the reason we choose them for making comparative analysis with the proposed heuristic approach.

In all the implementations, we have used same intersection, traffic and safety gaps. In the proposed scheme, we adapt the

vehicle length in simulation by adding a delay equal to the time that vehicle would take to pass the conflict point and depart point. Since vehicles travel only on the center of their respective lane, lateral safety gap is not required. Results are obtained for delay in vehicle trip time. Delay time is chosen as the parameter for performance evaluation because it is an independent parameter and other performance matrices such as mean velocity of vehicles etc. will have non-orthogonal dependence on it.

Values of parameters used in the simulation are given in Table 2.

TABLE 2. Parameter values.

Parameter	Value
Maximum Velocity	$17m/s$
Acceleration	$3m/s^2$
Safety gap	$500ms$
Approach length	$150m$
Buffer length	$50m$
Simulation time	$3600s$

TABLE 3. Simulation results.

	Delay (in Sec.)			
Density(v/h)	TL	FCFS	CIVIC	Proposed
500	13.88	0.288	1.012	0.0416
1000	14.56	0.615	1.234	0.059
1500	14.60	0.833	1.387	0.093
2000	15.83	1.604	1.500	0.149
2500	16.03	1.827	1.768	0.209
3000	16.68	2.430	1.845	0.236
3500	18.06	2.479	1.932	0.264
4000	18.87	3.077	1.985	0.328
4500	24.65	3.250	2.027	0.329
5000	29.02	4.090	2.125	0.389

Simulation is performed for different traffic densities for a simulation time of 3600 seconds. With similar environmental and traffic conditions, delay times are recorded for different schemes of intersection management. These are shown in Table 3 and visualized in Figure 6.

As we can see, the presented algorithm clearly outperforms other three intersection management algorithms. Traffic light have the largest delay owing to the fixed halt of vehicles during red light making vehicles to decelerate and again accelerate on green light causing delay in these transitions of velocity plus the wait time during the red phase. Our scheme outperforms FCFS algorithm because in FCFS algorithm, an availability check is performed at all the granules (blocks) which are lying in the trajectory of the vehicles. On the other hand, in our scheme, we only need to make 4 availability checks corresponding to the four conflict points. And the reason why the proposed algorithm results in less delay than

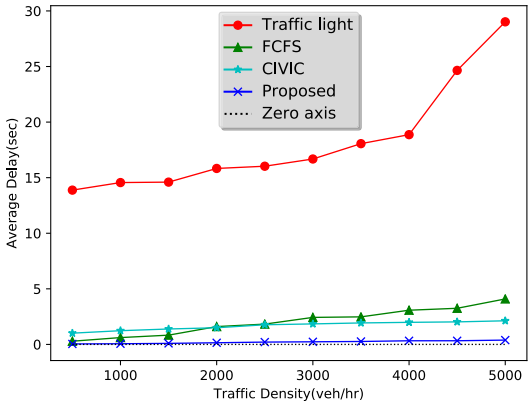

FIGURE 6. Comparison result.

CIVIC algorithm is that in CIVIC algorithm, optimization is performed only to minimize the overlap of trajectories of two vehicles inside the intersection area and no factor is added to this optimization that targets to reduce the delay in scheduling.

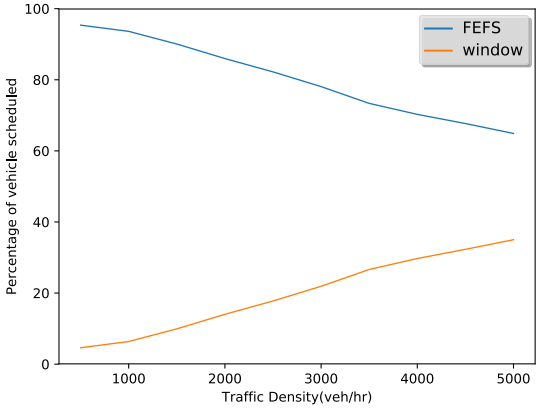

FIGURE 7. Percentage of vehicles scheduled by FEFS and Window algorithm.

Figure 7 shows the number of vehicles scheduled by FEFS and Window scheme as a function of traffic density. It gives an evidence of the fact that FEFS schedules most of the vehicles at a low traffic density whereas window algorithm finds more use at a higher traffic density. Reservation scheme is not shown in this figure because it's maximum contribution came out to be less than 0.1 percent of all the vehicles in the simulation.

VI. DISCUSSION AND FUTURE WORKS

The main contribution of the presented work is to obtain a collision free routing of vehicles incoming to an intersection by using a heuristic algorithm which is fast and suitable for real-time behavior of the task as it does not involve any computationally intensive procedure. The proposed heuristic is suitable for the IM task also because intersection management is an NP-Hard problem. In this work, along with scheduling vehicles inside the intersection area, their traversal inside the approach lane is also governed by the intersection manager. Finally, we performed a comparative

study against three different IM strategies namely, (i) Traffic light, (ii) FCFS scheme and (iii) CIVIC algorithm.

Results suggest how efficient the traffic management can be when it is totally composed of autonomous vehicles only. Even intersections, which are currently known as the 'Bottlenecks of traffic' will bring very little effect in the vehicle's trip.

In the presented work we have considered trajectory inside the intersection square to be fixed and vehicles are required to travel on specified lane only. These restrictions when lifted, will result in more complexity but will bring the model closer to the real world conditions. Future works will be targeted on lifting these restrictions and including manually driven vehicles as well in the scenario.

REFERENCES

[1] *Automated Driving Systems*. Accessed: Jun. 2018. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf

[2] *Autonomous Vehicle Information Library*. Accessed: Jun. 2018. [Online]. Available: https://www.aamva.org/Autonomous-Vehicle-Information-Library/

[3] *Vehicles Autonomy Levels*. Accessed: Jun. 2018. [Online]. Available: https://www.sae.org/binaries/content/assets/cm/content/news/press-releases/pathway-toautonomy/automated_driving.pdf

[4] A. Giridhar and P. R. Kumar, "Scheduling automated traffic on a network of roads," *IEEE Trans. Veh. Technol.*, vol. 55, no. 5, pp. 1467–1474, Sep. 2006.

[5] J. Garcia-Nieto, A. C. Olivera, and E. Alba, "Optimal cycle program of traffic lights with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 6, pp. 823–839, Dec. 2013.

[6] C. Vilarinho, J. P. Tavares, and R. J. F. Rossetti, "Intelligent traffic lights: Green time period negotiaton," *Transp. Res. Procedia*, vol. 22, pp. 325–334, Jan. 2017.

[7] O. Younis and N. Moayeri, "Employing cyber-physical systems: Dynamic traffic light control at road intersections," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2286–2296, Dec. 2017.

[8] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transp. Res. C, Emerg. Technol.*, vol. 85, pp. 732–752, Dec. 2017.

[9] J. L. Fleck, C. G. Cassandras, and Y. Geng, "Adaptive quasi-dynamic traffic light control," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 830–842, May 2016.

[10] S. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Vehicular networks for collision avoidance at intersections," *SAE Int. J. Passenger Cars-Mech. Syst.*, vol. 4, no. 1, pp. 406–416, 2011.

[11] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection management using vehicular networks," SAE Tech. Papers 2012-01-0292, Apr. 2012.

[12] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "V2V-intersection management at roundabouts," *SAE Int. J. Passenger Cars-Mech. Syst.*, vol. 6, no. 2, pp. 681–690, Apr. 2013.

[13] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, "Self-organized traffic control," in *Proc. 7th ACM Int. Workshop Veh. InterNETworking*, Sep. 2010, pp. 85–90.

[14] F. Zhu and S. V. Ukkusuri, "A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment," *Transp. Res. C, Emerg. Technol.*, vol. 55, pp. 363–378, Jun. 2015.

[15] F. Altché and A. de La Fortelle, "Analysis of optimal solutions to robot coordination problems to improve autonomous intersection management policies," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 86–91.

[16] G. R. de Campos, P. Falcone, R. Hult, H. Wymeersch, and H. Sjöberg, "Traffic coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 1, pp. 8–21, 2017.

[17] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1162–1175, Sep. 2013.

[18] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proc. 15th ACM Int. Conf. Hybrid Syst., Comput. Control*, Apr. 2012, pp. 145–154.

[19] K. L. Zhang, D. F. Zhang, A. de La Fortelle, X. Wu, and J. Grégoire, "State-driven priority scheduling mechanisms for driverless vehicles approaching intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2487–2500, Oct. 2015.

[20] F. Basma, Y. Tachwali, and H. H. Refai, "Intersection collision avoidance system using infrastructure communication," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 422–427.

[21] M. Elhenawy, A. A. Elbery, A. A. Hassan, and H. A. Rakha, "An intersection game-theory-based traffic control algorithm in a connected vehicle environment," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2015, pp. 343–347.

[22] P. Dai, K. Liu, Q. Zhuge, E. H.-M. Sha, V. C. S. Lee, and S. K. Son, "Quality-of-experience-oriented autonomous intersection control in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1956–1967, Jul. 2016.

[23] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *Proc. 3rd Int. Joint Conf. Auto. Agents Multiagent Syst.*, Jul. 2004, pp. 530–537.

[24] K. Dresner and P. Stone, "Multiagent traffic management: An improved intersection control mechanism," in *Proc. 4th Int. Joint Conf. Auton. Agents Multiagent Syst.*, Jul. 2005, pp. 471–477.

[25] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 529–534.

[26] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, Feb. 2016.

[27] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 81–90, Mar. 2012.

[28] I. H. Zohdy, R. K. Kamalanathsharma, and H. Rakha, "Intersection management for autonomous vehicles using iCACC," in *Proc. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Anchorage, AK, USA, Sep. 2012, pp. 1109–1114.

[29] A. Parker and G. Nitschke, "How to best automate intersection management," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1247–1254.

[30] D. Miculescu and S. Karaman, "Polling-systems-based control of high-performance provably-safe autonomous intersections," in *Proc. IEEE 53rd Annu. Conf. Decis. Control (CDC)*, Dec. 2014, pp. 1417–1423.

[31] D. Krajzewicz and J. Erdmann, "Road intersection model in SUMO," in *Proc. 1st SUMO User Conf.*, 2013, pp. 212–220.

AADITYA PRAKASH CHOUHAN (S'18) received the B.E. degree in electronics and instrumentation from the Institute of Engineering and Technology, Indore, India, and the M.Tech. degree in systems and control from the Electrical Department, IIT Roorkee, India. He is currently a Research Scholar with IIT Indore. His current research works involve combination of cyber physical systems, autonomous agents, and intelligent transportation systems.

GOURINATH BANDA (M'07) received the Ph.D. degree in computer science from Roskilde University and the M.S.Engg. degree in mechatronics from the University of Southern Denmark, Denmark. He is currently an Associate Professor in computer science and engineering with IIT Indore, India. In the past, he was a Chief Engineer with the Advanced Technology Group, Samsung Research and Development Labs, India, and a Scientist Fellow with the National Aerospace Laboratories, India. He has one U.S. patent, 2+ pending patent applications, and 10 technical publications. His research interests include developing (autonomous) cyber physical systems, real-time kernels design, and formal verification of real-time systems.