Vehicle-in-the-loop (VIL) Verification of a Smart City Intersection Control Scheme for Autonomous Vehicles

S. Alireza Fayazi  Ardalan Vahidi

Abstract—In this paper, an implementation of a traffic management scheme is presented for autonomous vehicles at intersections. Using bi-directional communication provided by a cellular network, the intersection controller is capable of receiving the status of the approaching vehicles, calculating an optimal arrival schedule, and sending the obtained schedule to individual vehicles. The proposed algorithm eliminates the need for physical traffic signals in an all autonomous driving environment. A vehicle-in-the-loop (VIL) simulation environment is implemented in order to evaluate our proposed traffic management scheme in reducing stops and energy consumption while ensuring safety. The test environment is configurable and reproducible to incorporate most of the real time signal control methods. This environment can assist the developers to validate their smart city projects.

I. INTRODUCTION

Autonomous vehicles can benefit from traffic signal information by precisely controlling their speed and arrival time at a green light. If all vehicles are autonomous, such as in a future smart city, then physical traffic light is not needed anymore as shown in concept papers by [1–6]. In such an environment, the intersection controller can rapidly switch between phases because autonomous-controlled vehicles have much faster reaction times than human-controlled vehicles [7]. However, the major challenges in developing a traffic management scheme for autonomous vehicles are first to formulate an intelligent intersection control that is responsive to prevailing traffic conditions, and second to create a versatile live testbed for intersection controllers.

We propose a novel intersection control scheme at the cyber layer to facilitate uninterrupted intersection passage for autonomous vehicles. The challenge is to provide vehicles with travel recommendations that ensure energy efficiency, safety, passenger comfort, and smoother traffic flow. Incorporating all these goals into the intersection control algorithm will lead to very complex formulations. However, by proposing an imaginary access area around the intersection, we managed to have a formulation based only on time of arrivals to that area. Then we converted our vehicle arrival scheduling problem to a Mixed Integer Linear Program (MILP), as described extensively in author’s recent work [8]. In this paper, the optimization problem is solved using IBM CPLEX solver on a cloud server and the corresponding outputs (scheduled access/arrival times) are sent to all approaching vehicles in an experimental testbed.

As mentioned earlier, developing an experimental testbed is another major challenge. A Vehicle-In-Loop (VIL) testbed seems a preferable strategy in early stages of a smart city development where cost and safety are a priority. A VIL configuration needs a microscopic traffic simulation model, one or more real vehicles, and a communication scheme between the intersection controller and all vehicles (simulated and real vehicles). The actual test vehicles at the physical layer interact with 1) simulated vehicles in a traffic microsimulation layer, and 2) intersection controller that run and interact on a remote back-end cloud. The simulated vehicles have to replicate the same communication protocol as the real vehicles. These requirements are considered in the VIL design presented in this paper.

To the best of our knowledge, to date, only the traffic signal microsimulation presented by Quinlan et al. [9] is also implemented in a vehicle-in-the-loop manner. Their intersection manager uses a reservation paradigm [1] which allows the vehicles to reserve a block in space-time in the intersection. Their solution is not optimal in the sense that it is based on the First Come, First Serve methodology, and a reservation is rejected if any part of the requested space-time block has been previously reserved or occupied by another vehicle. However, the current paper proposes an optimization-based approach for intersection traffic management. In addition, from the experimental point of view, this paper uses cellular network to establish a two way communication link with a remotely situated control server, adds a fuel rate tracker to the VIL platform, and reports fuel efficiency benefits compared with baseline road tests.

First, our VIL simulation platform is explained in Section II. Our proposed access area and MILP-controlled intersection are explained in Section III. For completeness, our intersection scheduling algorithms presented in [8] is briefly described in Section IV, followed by our fuel estimation tool in Section V. The subsequent sections explain other components of our VIL setup including our virtual driver assistant, and micro-simulation engine (Section VI, and VII). The implemented semi-virtual testbed, and the benchmarking method as well as the experimental results are presented in Sections VIII, IX, and X followed by our conclusion.

Seyed Ali Reza Fayazi (sfayazi@g.clemson.edu), and Ardalan Vahidi (avahidi@clemson.edu) are with the Department of Mechanical Engineering, Clemson University, Clemson, SC 29634-0921, USA.
II. VEHICLE-IN-THE-LOOP CONFIGURATION

We add real vehicles to our microsimulations at an intersection testbed. Simulated and real vehicles all send and receive data in the same structure and format (see Figure 1). The proposed approach addresses many limitation of a simulation-only environment, while also ensuring a safe environment for test vehicles. In our vehicle-in-the-loop platform, conflicting movements (and potential crashes) occur in a virtual environment. Figure 2(a) shows a screenshot of the real vehicle interacting with hundreds of simulated vehicles. We mounted our simulator node inside a test vehicle as shown in Figure 2(b). Each test vehicle is equipped with a fuel rate tracker and a virtual driver assistant, implemented as iOS applications (see Figure 1 and 2(b)).

A virtual traffic light is also embedded into our virtual driver assistant for our baseline testbed and is described later in Section IX.

III. PROPOSED INTERSECTION

Ignoring all the turns to simplify the presentation of ideas, we assume a square two-phase/four-movement intersection with width $W = 10$ m. As shown in Figure 3, we consider a two-phase intersection consisting of Phase X and Phase O as $\phi = \{\phi_X, \phi_O\}$. Each phase is allocated to one or more non-conflicting movements. The set of all vehicular traffic movements used in this paper is denoted by $M$ (see Figure 3).

For each intersection, we will assume a subscription process by which the approaching connected vehicles send subscription requests to the intersection control server and announce their presence as well as their intended time of arrival. We represent the list of all subscribed connected vehicles as $CV = \{cv_i\}_{i=1}^n$ where $n$ is the size of $CV$. The list of connected vehicles is sorted by distance to the intersection where $cv_1$ is the closest vehicle to the intersection.

For each vehicle approaching an intersection, we are interested in the following time instances: (1) time when the front of the vehicle enters the intersection area at the stop-bar; (2) time when the rear of the vehicle exits the intersection area; (3) time when the front of the vehicle reaches an access distance from the intersection. As shown in Figure 3, these time instances are denoted by $t_{enter}$, $t_{exit}$, and $t_{access}$, respectively. In this figure, the intersection and access areas are shown by a shaded area and a solid box, respectively. The access area border is defined by $d_{access}$, that is the estimated stopping distance of a vehicle in case of a safety concern and is calculated as a function of the road average speed $v_{avg}$:

$$d_{access} = t_{res}v_{avg} + \frac{0 - v_{avg}^2}{2a_{dec,max}} \tag{1}$$

where $t_{res} = 0.5$ sec is assumed to be the response time of an autonomous vehicle, and $a_{dec,max} = -4$ m/s$^2$ is the maximum deceleration considered for passenger cars when a dangerous situation is detected. We obtain $d_{access} \approx 38$m by setting $v_{avg} = 56.3$ kph (35 mph). It should be emphasized that intersection access time ($t_{access}$) for each vehicle is the time that vehicle enters the access area; all other vehicles in the opposing movement must access the intersection at a sufficiently later time.

The attributes of each vehicle $cv_i \in CV$ (1 \leq i \leq n) subscribed to an intersection controller are described by:

$$cv_i = \langle m_i, \phi_i, d_i, v_i, t_{access,i}, t_{access,des,i}, t_{exit,i} \rangle \tag{2}$$

where $m_i$ is the vehicle movement $m_i \in M$, $\phi_i$ is the phase $\phi_i \in \phi$ that $cv_i$ movement is associated with, $d_i$ is the distance of $cv_i$ to the intersection access point, $v_i$ is the velocity of $cv_i$, $t_{access,i}$ is the assigned time-stamp for $cv_i$ to access the intersection, and $t_{access,des,i}$ is the $cv_i$’s desired access time. Please note that in this paper, we assume that all vehicles
prefer to travel at the average velocity \( v_{avg} \); and as a result, their distance divided by \( v_{avg} \) yields their desired access times with respect to current time \( t_0 = 0 \) sec.

While a vehicle is approaching an intersection, it first sends a subscription request and announces its intended time of arrival. An unsubscribe message is later sent from the vehicle to the intersection controller server at the time the vehicle clears the intersection. Thus, data is exchanged only during the subscription period. The data exchanged during this period is summarized in Figure 4. We exchange the information through a User-Datagram Protocol or UDP unconnected datagram sockets.

IV. MILP-BASED INTERSECTION CONTROLLER

The intersection controller resides on a server and receives information of all subscribing vehicles and then schedules the intersection access time for each vehicle regularly. The scheduled access-times (arrival-times) are sent to all subscribing vehicles so that they can adjust their speed accordingly. The challenge is to find appropriate access-times that ensure safety, passenger comfort, and less traffic delay. In this section, we show that it is possible to find such access-times via an optimization problem. Our linear objective functions and mixed integer linear constraints, presented in the following subsections, make the problem a Mixed Integer Linear Program (MILP).

A. Objective

The objective of increasing intersection throughput will be formalized here as an optimization problem. The main goal is to find the optimal sequence and time of arrival \( t_{access} \) for each vehicle such that the difference between the current time \( t_0 \) and the expected arrival time of the last vehicle (furthest subscribed vehicle) passing the intersection in a given time window is minimized. This objective will maximize the number of vehicles that clear the intersection in a given time:

\[
J_1 = t_{access,n} - t_0
\]

s.t. \( n = \#CV \)

\( t_{access,n} \geq \{t_{access,1}, \ldots, t_{access,n-1}\} \)

Minimizing the aforementioned objective could force the vehicles to travel near the speed limit against their preference. To avoid such a scenario, we incorporate the desired arrival time of the vehicles into the optimization problem in such a way that vehicles would not face extreme delay or expedition compared to their desired arrival times. In other words, we define a cost on the difference between assigned and desired access times for all vehicles:

\[
J_2 = \sum_{i=1}^{n} |t_{access,i} - t_{access,des,i}|
\]

Equation (4) is not linear and needs to be restated. We restate \( J_2 \) by adding a new so-called slack variable \( \Delta t_{access,abs,i} = |t_{access,i} - t_{access,des,i}| \). Then, considering the fact that we are minimizing the cost function and \( |x| = \max\{x, -x\} \) for any real number \( x \), we can add two constraints given in Equation 5 in order to ensure that our added slack variable is equal to \( |t_{access,i} - t_{access,des,i}| \):

\[
\begin{align*}
J_2 &= \sum_{i=1}^{n} \Delta t_{access,abs,i} \\
\text{s.t.} \quad \Delta t_{access,abs,i} &\geq (t_{access,i} - t_{access,des,i}) \\
\Delta t_{access,abs,i} &\geq -(t_{access,i} - t_{access,des,i})
\end{align*}
\]

The total cost function to be minimized is then:

\[
J = w_1 J_1 + w_2 J_2
\]

where \( w_1 \) and \( w_2 \) are penalty weights. We hypothesize that this optimization will result in reduced fuel consumption and intersection delay, even though these factors are not explicitly incorporated into the objective function. The results presented in Section X are obtained by setting \( w_1=80\% \) and \( w_2=20\% \).

B. Constraints

1) Speed limit and maximum acceleration: For each vehicle \( cv_i \), we should consider the speed limit requirement \( v_i \leq v_{max} \) as well as the maximum acceleration constraint \( a_i \leq a_{acc,max} \); where \( v_i \) and \( a_i \) are the velocity and acceleration of the vehicle, \( v_{max} \) is set based on the speed limit of the road, and \( a_{acc,max} \) is set to \(+3 \) m/s\(^2\). We introduce \( t_{access,\text{min,i}} \) as the earliest time that \( cv_i \) can access the intersection, if it travels with maximum acceleration and speed possible. Then we rephrase our aforementioned speed and acceleration constraints as:

\[
t_{access,i} \geq t_{access,\text{min,i}}
\]

2) Safety gap on the same movement: Two consecutive vehicles that are traveling on the same movement (e.g. east bound) should be separated by a safety gap (headway) that is denoted by \( t_{gap1} \) in this paper. This time gap is independent of the vehicles’ speed [3] (except at very low speeds), and is the minimum following time gap to avoid a rear end collision. As suggested in [3], a 1 sec headway provides a reasonable upper bound for an autonomous vehicle response time. However, a 1 sec headway is not sufficient at very low speeds such as when discharging from a queue. We observed in simulations that when discharging from a queue,
the headway between the first vehicle and the second vehicle can be as large as 2.3 sec. As a result, we default to $t_{gap1} = 2.5$ sec in our optimization formulations; only if it is determined that a vehicle will access intersection at a large enough speed we set $t_{gap1} = 1$ sec. To enforce the headway, we add the following constraint on any two consecutive vehicles traveling on the same movement:

$$t_{access,j} - t_{access,k} \geq t_{gap1}$$

**s.t.** $cv_j, cv_k \in CV; \quad d_j \geq d_k$; $m_j, m_k \in M; \quad m_j = m_k.$

(8)

3) **Safety gap on different movements:** Two vehicles traveling on different phases (conflicting movements) also need to be separated by a safety gap. This time gap, if selected properly, guarantees that a vehicle can only enter the access area after all conflicting vehicles have left the intersection area. Considering two vehicles $cv_j$ and $cv_k$ that are on different phases of $\phi_j \in \Phi$ and $\phi_k \in \Phi (\phi_j \neq \phi_k)$, the following constraints cover all the possible situations with just enough safety gap between the vehicles; here $\lor$ is the OR operator:

$$t_{access,j} - t_{exit,k} \geq 0$$

$$t_{access,k} - t_{exit,j} \geq 0$$

**s.t.** $cv_j, cv_k \in CV; \quad \phi_j, \phi_k \in \Phi; \quad \phi_j \neq \phi_k.$

(9)

We are specifically interested in the time gap between accessing timestamps so that, at the end, we can derive a simple linear constrained optimization problem based on access times only. For this reason, we define $t_{exit} = t_{access} + \Delta_{travel}$ where $\Delta_{travel}$ is the travel time between access point and exit point of a vehicle. Now, by substituting $t_{exit} = t_{access} + \Delta_{travel}$ into Constraint (9), we can rephrase this constraint as:

$$t_{access,j} - t_{access,k} \geq t_{gap2}$$

$$t_{access,k} - t_{access,j} \geq t_{gap2}$$

**s.t.** $cv_j, cv_k \in CV; \quad \phi_j, \phi_k \in \Phi; \quad \phi_j \neq \phi_k.$

(10)

where $t_{gap2} = \Delta_{travel}$ is the safety gap we need between access times. The travel time between the access point and the exit point of a vehicle is equal to the time period it needs to first pass the access distance ($d_{access}$), then pass the intersection area, and finally exit the intersection completely. The longest travel time a vehicle could take is when it is stopped behind the access area and accelerates at its assigned access time. We set an average acceleration rate of 2 m/s² for the vehicles and obtain $\Delta_{travel} = 7.3$ sec as the longest travel time. Consequently, we set $t_{gap2} = 7.5$ sec.

We need to convert Constraint (10) into an AND-combination of two or more equations in such a way that if one equation holds true then the other equations are always redundant. The most widely known method to handle this is the big-M method that, in our application, requires a binary variable $B_i$ and a constant $M_{big}$ [10]. For each set of Constraint (10) applying on two vehicles of $cv_j$ and $cv_k$, we add one artificial binary variable $B_i (1 \leq i \leq \# of constraints)$ to take care of the discontinuity as:

$$t_{access,j} - t_{access,k} + M_{big} B_i \geq t_{gap2}$$

$$t_{access,k} - t_{access,j} + M_{big} (1 - B_i) \geq t_{gap2}$$

**s.t.** $cv_j, cv_k \in CV; \quad \phi_j, \phi_k \in \Phi; \quad \phi_j \neq \phi_k; \quad B_i \text{ binary}$

(11)

where $B_i$ can be either 0 or 1, $M_{big}$ is a large enough number, and $\land$ is AND operator.

V. FUEL USAGE TRACKER

We developed an iOS application that logs the vehicle On-Board Diagnostics (OBD) data as well as the iOS device sensor data. As shown in Figure 5, the OBD Log application can connect to commercial Wi-Fi OBDII readers supporting ELM327 [11]. The application uses the OBD data to identify the changes in engine’s mode of operation, and estimate the instantaneous fuel consumption rate of the engine.

![Fig. 5: Functional architecture of the developed iOS OBD Logger App.](image)

Some vehicles provide enhanced OBD data which includes the engine’s fuel rate estimate; however, this OEM-specific data needs to be first evaluated, and may not be accurate enough for verification purposes of this paper. Furthermore, not all vehicles, like our test vehicle, provide the fuel consumption rate as an OEM-specific parameter. What we implemented here is a method that estimates the engine fuel consumption with a mean absolute error of 1.0% using only basic OBD data [12].

VI. VIRTUAL DRIVER ASSISTANT

As no autonomous vehicle was available, we used conventional vehicles as our test vehicles and we guided our test drivers via a virtual driver assistant so that they can follow the planned speed. The virtual driver assistant displays the appropriate speed recommendation to the driver as green zones on a GPS speedometer (please see Figure 2(b)). We developed this application for iOS devices (iPhone, or iPad) and the designed human machine interface (HMI) is explained in Figure 6.

The implemented driver assistant is a location-based application and it is able to determine which intersection and which phase/movement is relevant to a vehicle along a trip. Once the vehicle enters a monitoring region around the intersection, the virtual driver assistant sends a subscription
request to the intersection controller. The MILP-based intersection controller, in return, sends the assigned access time. Based on the distance of the vehicle to the intersection, the appropriate speed to be followed by the driver is computed by a trajectory-planning engine embedded in the driver assistant. The computed speed is then displayed to the driver as green zones on the speedometer (adopted from [13], [14]), as seen in Figure 6. The goal is to guide the driver for a timely arrival at intersection in almost the same way that a real autonomous vehicle would do.

VII. TRAFFIC MICROSIMULATION

The simulation tool, developed in this work, models the autonomous vehicles as agents that decide their travel trajectory by their individual trajectory-planners. A two-phase/four-movement intersection is simulated, as previously shown in Figure 3. Each intersecting road has one lane per direction, and no turning is allowed at the intersection. The microsimulations were implemented using Java.

VIII. SEMI-VIRTUAL SMART CITY TESTBED

A test track at International Transportation Innovation Center (ITIC) [15] in Greenville, South Carolina was used to validate the proposed intersection control scheme in a vehicle-in-the-loop platform. The ITIC testing infrastructure is a part of South Carolina Technology & Aviation Center (SCTAC) and provides a 600-acre closed test site. We drove our test vehicle on a 5,500 x 300-foot isolated asphalt straightaway located at ITIC test area and shown on satellite view in Figure 7. The test vehicle was a Honda Accord LX 2.4L 4-Cylinder SI gasoline engine.

As shown in Figure 7, an imaginary intersection was set up using traffic cones. A stop sign was also placed at the access area border so that the test driver would stop at that location if a stop was required by the virtual driver assistant. The MILP-based intersection control server was located remotely from the testbed. The real vehicles interacted with the intersection control cyber-layer and with the microsimulations in a virtual road network environment. The simulated vehicles arrive at three approaches at rates of 750 vehicles/hour using a stochastic generation method (negative exponential distribution [16]). Figure 2(a) demonstrates that one approach (O') is dedicated to real vehicles only so that the simulated environment does not need to be visualized to the test driver while she/he is driving the real vehicle.

IX. BENCHMARKING

Autonomous vehicles approaching a pre-timed traffic signal control provide a baseline testbed, against which we can benchmark and compare our MILP-based intersection controller. The signal timing for the benchmark pre-timed traffic signal was obtained off-line from SYNCHRO (Trafficware 2011) optimization program. This optimized timing was then used in the microsimulation to model the current state of the traffic light (cycle time = 100 sec, green split = 44.5 sec, and yellow interval = 3.5 sec). The simulated vehicles’ arrival pattern at this baseline test was recorded and replayed later for our MILP-controlled intersection simulations. In this way, the same arrival pattern was exactly replicated for each test.

In this baseline testbed (namely Testbed Pre-Timed), we assumed that the autonomous vehicles were equipped with camera-based traffic signal state detection. This means that vehicles could only observe the current state of the signals ahead. In order to model this test environment, we set our programs in such a way that the simulated and real vehicles received traffic signal status information only when they were within the range of their imaginary camera. We set this range as 300 m measured from the intersection ahead. A virtual traffic signal was implemented to display the signal status to test driver as shown in Figure 6.

X. VIL SIMULATION RESULTS

Our MILP-based intersection control was tested with the pre-timed intersection benchmark in a vehicle-in-the-loop configuration (VIL). A total of two vehicle-in-the-loop simulation tests were conducted for Testbed Pre-Timed and Testbed MILP. The average and maximum speeds were set to \( v_{avg} = 56.3 \) kph (35 mph) and \( v_{max} = 72.4 \) kph (45 mph), respectively. The subscription distance was 1.2 km. Each test consisted of 12 laps around the test track with wide U-turns at both ends of the track. After each lap, there was a different period of rest (between 10 sec and 60 sec) so that the obtained results would not be affected by the cyclic periodicity of the pre-timed intersection benchmark.

The results of the VIL simulations for each testbed are given in Table I. In Testbed MILP, the mixed-integer linear programming problem was solved by IBM’s CPLEX optimization package. The measures of effectiveness (MOEs) studied in our VIL simulations and given in Table I are: (1) the intersection total number of stops, (2) the intersection...
The proposed technology can be utilized in smart city projects where only autonomous vehicles are allowed to travel. However, it might be possible to modify the proposed algorithm for a mixed traffic consisting of autonomous and human controlled vehicles. In a mixed traffic environment, a physical traffic light is needed. Other future works include examining the robustness of our algorithms and identifying the situations where no solution exists.

**ACKNOWLEDGMENT**

This research was partially sponsored by a research award from BMW Information Technology Research Center (ITRC) in Greenville, SC. The authors are thankful for the support provided by Dr. Andre Luckow from BMW ITRC. Some of the ideas presented in this paper started to take shape when the authors were working at BMW Technology Office USA in Mountain View, CA. The authors would like to thank Mr. Andreas Winckler from that office. They also thank Dr. Joachim Taiber of International Transportation Innovation Center (ITIC) for providing the authors with access to a closed test site.

**REFERENCES**


