Cooperative Schedule-Driven Intersection Control with Connected and Autonomous Vehicles

Hsu-Chieh Hu[1], Stephen F. Smith[2], Rick Goldstein[2]

***Abstract*— Recent work in decentralized, schedule-driven traffic control has demonstrated the ability to improve the efficiency of traffic flow in complex urban road networks. In this approach, a scheduling agent is associated with each intersection. Each agent senses the traffic approaching its intersection and in real-time constructs a schedule that minimizes the cumulative wait time of vehicles approaching the intersection over the current look-ahead horizon. In this paper, we propose a cooperative algorithm that utilizes both connected and autonomous vehicles (CAV) and schedule-driven traffic control to create better traffic flow in the city. The algorithm enables an intersection scheduling agent to adjust the arrival time of an approaching platoon through use of wireless communication to control the velocity of vehicles. The sequence of approaching platoons is thus shifted toward a new shape that has smaller cumulative delay. We demonstrate how this algorithm outperforms the original approach in a real-time traffic signal control problem.**

I. Introduction

Traffic congestion has been becoming an increasingly critical problem and is getting worse due to population shifts to urban areas and the high usage rate of vehicles. Traffic jams are now common on urban surface streets, where It is commonly recognized a better optimization of traffic signals could lead to substantial reduction of traffic congestion. A recent development in decentralized online planning to traffic signal control problem has achieved significant improvements to urban traffic flows through real-time, distributed generation of long-horizon, signal timing plans. [1], [2] The key idea behind this *schedule-driven* approach is to formulate the intersection scheduling problem as a single machine scheduling problem, where input jobs are represented as sequences of spatially proximate vehicle clusters (approaching platoons, queues). This aggregate representation enables efficient generation of long horizon timing plans that incorporate multi-hop traffic flow information, and thus network-wide coordination is achieved through exchange of schedule information among neighboring intersections.

Within the schedule-driven traffic control approach, a generated signal timing plan is executed in rolling horizon fashion, and is recomputed every second or so to account for uncertain traffic conditions. Each time a timing plan is regenerated, the instantaneous state of approaching cluster sequences is used as a predictive model, to provide a tractable estimation of current demand that preserves the non-uniform nature of real-time traffic flows. This information, which includes the vehicles that comprise each cluster and their respective arrival times at the intersection, not only provides a basis for generating signal timing plans that optimize the actual traffic on the road, but also paves a way to further optimize flows through collaboration between infrastructure (the intersection scheduling agents) and vehicles.

The recent development of connected and autonomous vehicle (CAV) technologies (such as vehicle-to-infrastructure (V2I) communication and autonomous vehicle control), provides opportunities to improve the driver's behavior, and, in the longer term, to even control the movement of vehicles, for purposes of improving the efficiency and safety of transportation. The emergence of autonomous vehicle control enables rapid agile response to the traffic information sent by the traffic signal system, which could not necessarily be expected for human drivers. Hence, the availability of such technologies provides new flexibility to the design of new traffic control systems. On the other hand, the rich traffic information that will be made available by V2I communication technologies can enable the development of better traffic control strategies for optimizing a given performance objective, e.g, delay or capacity (throughput).

In this paper, we propose an algorithm that achieves improved traffic flow efficiency through collaboration between schedule-driven traffic control and CAVs. The proposed algorithm takes advantage of the detailed real-time timing information provided by schedule-driven traffic control and the availability of autonomous control (or a driver advisory system) through CAVs to establish a framework for interaction. At the beginning of each planning cycle, the intersection scheduling agent first generates a signal timing plan based on the detected traffic flow as before. Then, based on this timing plan, the agent determines a set of velocities for all approaching vehicles (essentially speeding up and slowing down selected approaching vehicles) that enables generation of a better signal timing plan. After receiving its suggested speed, each approaching vehicle controls its speed accordingly. The cluster representation is adjusted through iterative application of two operations, each of which involve merging and splitting of existing clusters: (1) Re-sequencing vehicles belonging to different traffic flows (i.e., belonging to different signal phases) by schedule-driven traffic control, and (2) Adjusting the speed of approaching vehicles and intersections such that the gaps between clusters competing

*This research was funded in part by the University Transportation Center on Technologies for Safe and Efficient Transportation at Carnegie Mellon University and the CMU Robotics Institute.

[1]Hsu-Chieh Hu is with Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. hsuchieh@andrew.cmu.edu

[2]Stephen F. Smith and Rick Goldstein are with the Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. {sfs, rgoldste}@cs.cmu.edu

for green time are enlarged. It is shown that, in scenarios with heavy congestion, the cooperative method outperforms schedule driven control, resulting in an additional delay reduction of 19%. Furthermore, this algorithm is still feasible in the case that the penetration rate of CAV is low.

The remainder of the paper is organized as follows. We first introduce related work and schedule-driven traffic control. Next, the algorithm necessary to achieve cooperation and its theoretical analysis are discussed. Then, an empirical analysis of the composite approach is presented. Finally, some conclusions are drawn.

II. Related Work

Traditionally, there are three general approaches to control traffic signal: a) fixed timing; b) actuated; and c) adaptive. The earliest implementations are based on a fixed timing method optimized using historical traffic data offline. The later advancements have used actuated or adaptive signals. Then, if all cars are equipped with wireless communication technologies, e.g., Dedicated Short Range Communications (DSRC), to communicate with a centralized infrastructure, we can optimize the traffic flow by ordering the phases of traffic signal more efficiently. In [3], [4] information from equipped vehicles is used to determine demands and optimize the cycle length and green splits of a traffic signal once every cycle. In [5] the presence of platoons is detected using V2I communication and a mixed integer non-linear program is solved to produce the optimal phasing sequence. However, this approach does not scale well for generating long horizon plans and does not deal with uncertainty of traffic states.

Traffic flow can also be optimized by searching for the optimal discharge sequence for each individual vehicle. One possible approach is request-based framework where each vehicle reserves a space through sending request in advance. For example, [6] proposed a novel intersection control method called Autonomous Intersection Management (AIM), and in particular described a First Come, First Served (FCFS) policy to direct autonomous vehicles when to pass through the intersection. They showed that by leveraging the capacity of computerized driving systems FCFS significantly outperforms traditional traffic signals. Another similar approaches is [7] in which the system is slot-based and able to double capacity. Although those approaches are promising, they assume perfect penetration of connected vehicles and may not be realized in near future. We need a more practical approach to bridge between traditional methods (i.e., traffic signals) and request-based methods.

Taking advantage of autonomously controlled vehicles, and the use of information from connected vehicles for intersection control has been investigated in several researches. The trajectory of fully autonomous vehicles can be manipulated to optimize an objective function [8], [9], [10], [11]. Those approaches can achieve either better safety or efficiency through interaction between intersections and vehicles. [12] propose an extension of AIM to enable vehicles to apply motion planning for optimizing speed.

III. Schedule-Driven Traffic Control

As indicated above, the key to the single machine scheduling problem formulation of the schedule-driven approach of [1] is an aggregate representation of traffic flows as sequences of clusters c over the planning (or prediction) horizon. Each *cluster* c is defined as $(|c|, arr, dep)$, where $|c|$, arr and dep are number of vehicles, arrival time and departure time respectively. Vehicles entering an intersection are clustered together if they are traveling within a prespecified interval of one another. The clusters become the jobs that must be sequenced through the intersection (the single machine). Once a vehicle moves through the intersection, it is sensed and grouped into a new cluster by the downstream intersection.The sequences of clusters provide short-term variability of traffic flows at each intersection and preserve the non-uniform nature of real-time flows. Specifically, the *road cluster sequence* $C_{R,m}$ is a sequence of $(|c|, arr, dep)$ triples reflecting each approaching or queued vehicle on entry road segment m and ordered by increasing arr. Since it is possible for more than one entry road to share the intersection in a given *phase* (a phase is a compatible traffic movement pattern, e.g., East-West traffic flow), the *input cluster sequence* C can be obtained through combining the road cluster sequences $C_{R,m}$ that can proceed concurrently through the intersection. The travel time on entry road m defines a finite horizon (H_m), and the prediction horizon H is the maximum over all roads.

Every time the cluster sequences along each approaching road segment are determined, each cluster is viewed as a non-divisible job and a forward-recursion dynamic programming search is executed in a rolling horizon fashion to continually generate a phase schedule that minimizes the cumulative delay of all clusters. The frequency of invoking the intersection scheduler is once a second, to reduce the uncertainty associated with clusters and queues. The process constructs an optimal sequence of clusters that maintains the ordering of clusters along each road segment, and each time a phase change is implied by the sequence, then a delay corresponding to the intersection's yellow/all-red changeover time constraints is inserted based on Algorithm 1. If the resulting schedule is found to violate the maximum green time constraints for any phase (introduced to ensure fairness), then the first offending cluster in the schedule is split, and the problem is re-solved.

Formally, the resulting *control flow* can be represented as a tuple (S, C_{CF}) shown in Figure 1, where S is a sequence of phase indices, i.e., $(s_1, \cdots, s_{|S|})$, C_{CF} contains the sequence of clusters $(c_1, \cdots, c_{|S|})$ and the corresponding starting time after being scheduled. More precisely, the delay that each cluster contributes to the cumulative delay $\sum_{k=1}^{|S|} d(c_k)$ is defined as

$$d(c_k) = |c_k| \cdot (ast - arr(c_k)), \tag{1}$$

where ast is the actual start time that the vehicle is allowed to pass through, which is determined by Algorithm 1. Note that ast is determined by the arr and permitted start time

(pst) described in Algorithm 1. For a partial schedule S_k, the corresponding state variables are defined as a tuple, (s, pd, t, d), where s is phase index and pd is duration of the last phase, t is the finish time of the kth cluster and d is the accumulative delay for all k clusters. The state variable of S_k can be updated from S_{k-1}. Algorithm 1 is based on a greedy realization of planned signal sequence, where $MinSwitch(s, i)$ returns the minimum time required for switching from phase s to i and slt_i is the *start-up lost time* for clearing the queue in the phase i. We can use t and $MinSwitch(s, i)$ to derive pst. The optimal sequence (schedule) C^*_{CF} is the one that incurs minimal delay for all vehicles.

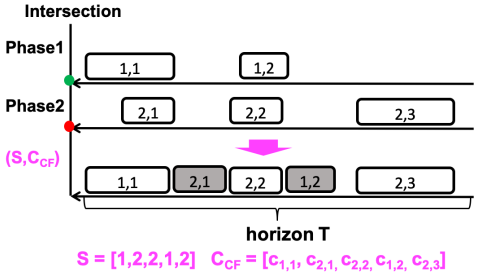

Fig. 1. The resulting control flow (S, C_{CF}) calculated by scheduling agents: each block represents a vehicular cluster. The shaded blocks represent the delayed clusters.

Algorithm 1 Calculate (pd, t, d) of S_k

Require: 1) (s, pd, t, d) of S_{k-1} ; 2) s_k

1: $i = s_k$; $c =$ next job of phase i
2: $pst = t + MinSwitch(s, i)$ ▷ Permitted start time of c
3: $ast = \max(arr(c), pst)$ ▷ Actual start time of c
4: **if** $s \neq i$ and $pst > arr(c)$ **then** $ast = ast + slt_i$
5: $t = ast + dep(c) - arr(c)$ ▷ Actual finish time of c
6: **if** $s \neq i$ **then** $pd = t - pst$
7: **else** $pd = pd + (t - pst)$
8: $d = d + |c| \cdot (ast - arr(c))$ ▷ Total accumulative delay
9: **return** (pd, t, d)

IV. Cooperative Algorithm

In this section, we introduce an algorithm that enables scheduling agents to assign velocities to the approaching vehicles through V2I communication. The goal is to further reduce cumulative delay compared to the baseline case above, where each vehicle passively follows timing plans at their free flow speeds. In brief, each scheduling agent computes a schedule every second. Based on the current cluster model and arrival prediction at any instant, the earliest time that any given vehicle can feasibly access the corresponding intersection (called the permitted starting time) can be determined by solving a single-machine scheduling problem to minimize cumulative delay. With these permitted times, the speeds of vehicles are adjusted by the scheduling agent through communication, and then the platoons are shifted to a new shape that incurs smaller cumulative delay. Basically, we revise the cluster representation through two operations: a) re-sequencing vehicles to minimize delay, and b) sending control message to adjust speed of vehicles.

A. Scheduling Information

The schedule generated by the scheduling agent contains useful timing information including the permitted start time (pst), actual start time (ast), phase time (pd) and arrival time (arr), etc. The approaching vehicles are able to apply these information to change their movement in order to achieve better traffic flow that has smaller delay. The original schedule-driven traffic control uses this information to control traffic signal timing and thus control vehicles in an indirect way. However, as CAV technologies are incorporated, the timing information allows the intersection to control vehicles directly through V2I communication. For instance, with $pst(c_i)$, we know the earliest time to cross the current intersection for cluster c_i. Each vehicle could check their $arr(c_i)$ to decide whether to speed up or slow down under a pre-defined safety constraint. If $arr(c_i) > pst(c_i)$, the vehicles of c_i could speed up to decrease the corresponding finish time t. On the other hand, if $arr(c_i) \leq pst(c_i)$, it is beneficial for the first several vehicles of c_i to maintain a lower speed and merge to the platoons from behind.

B. Optimizing Schedule via Changing Platoons

After acquiring the pst and arr of each cluster from the produced schedule, the scheduling agent starts to query the approaching vehicles about their current speed for improving delay further. We propose a greedy algorithm to improve the schedule with the speed information by iterating through each cluster and computing a new velocity that either shortens the corresponding phase or increases its crossing speed. First, we check whether the current cluster is a beginning of a new phase in Algorithm 2. If it is a new phase, we record the $pst_p = pst(c_i)$ for computing reduction δ of previous phase length, where p is index of phase and pst_p is the starting time of p_{th} phase. We consider two cases to compute δ:

1) If it is not that all previous vehicles need to wait for green time, δ is equal to the difference between travel time and the updated travel time of the last vehicle in previous phase, where the computation of the updated travel time will be described below.
2) If $pst_p - \delta \leq pst_{p-1}$ then the previous phase is too short, we need to move current phase forward and set $\delta = 0$. After computing δ, we know if the current phase is able to start earlier.

We maintain two variables, *end* and *updated_end*, to record the finish time of the previous phase before and after changing speed. *end* is updated with $arr(c_i)$ in Line 14, and *updated_end* is updated based on the new speed computed in Line 15 by Algorithm 3. The δ used for moving the phase finish time forward can be calculated from the difference between *end* and *updated_end*. Other than *updated_end*, the new speed in the control message that is sent is also computed by Algorithm 3. Algorithm 3 is based on $pst(c_i)$

Algorithm 2 Splitting clusters according to previous schedule

1: Apply forward recursion process
2: Retrieve the schedule solution (S^*, C^*_{CF}).
3: Query all vehicles about their current velocity v_i.
4: **for** $i = 1$ to C^*_{CF} **do**
5: Get $pst(c_i)$ and s_i
6: **if** $s_i \neq phase$ **then** ▷ new phase start
7: $pst_p = pst(c_i)$; $phase = s_i$; $\delta = 0$
8: **if** *end* $> pst_{p-1}$ **then** ▷ not all vehicles wait
9: $\delta =$ *end* $- \max(pst_{p-1},$ *updated_end*$)$
10: **if** $pst(c_i) - \delta \leq pst_{p-1}$ **then**
11: $\delta = 0$ ▷ δ is not too large.
12: *end* $=$ *updated_end* $= 0$;$p = p + 1$
13: $pst(c_i) = pst(c_i) - \delta$ ▷ update permitted start time
14: *end* $= \max($*end*$, arr(c_i))$
15: Compute v'_i and *updated_end* by Algorithm 3.
16: **if** *is_safe*(v'_i) **then**
17: Send speed control message to all vehicles in c_i

and $arr(c_i)$, where t_c in Line 1 is current time, Line 2 defines a ratio γ that is current travel time over the updated travel time, and the threshold in Line 3 is used for focusing on those critical vehicles and avoiding unnecessary speed change to lower communication overhead. If deviation of γ from 1 is large enough, we will update the speed according to a speed planning function *new_speed*$(\cdot)$, where *new_speed*$(\cdot)$ is a function that assigns speed for each vehicle based on current velocity v_i and γ. A suitable function based on the Intelligent Driver Model (IDM) [13] is

$$v'_i = v_i + a_{max}(1 - \gamma^{-\omega} - (\frac{s^*}{s})^2) \approx v_i + a_{max}(1 - \gamma^{-\omega}) \quad (2)$$

where a_{max} is the maximum acceleration and ω is acceleration exponent. We assume the current headway s^* is much larger than desired headway s, so that we can ignore the correction term. On the other hand, those vehicles with smaller deviation of γ from 1 are at the original speed for maintaining headway and space between vehicles. As we have the new speed, *updated_end* is updated in order to compute δ for modifying schedule. Figure 2 shows the three steps involved in changing a platoon sequentially and continually in a real-time manner.

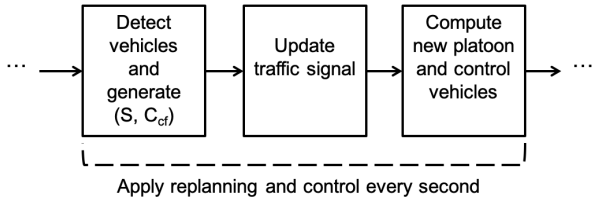

Fig. 2. The replanning and control cycle

Based on Algorithm 2, we can either speed up or slow down vehicles to improve the phase schedule (i.e., timing plan). As the vehicles speed up, it provides more space for the rear vehicles and makes the platoon more compact. For slowing down, it helps the approaching vehicles to keep a higher speed when crossing intersections. We prove that by applying Algorithm 2, the cumulative delay is guaranteed to be less than the previous schedule. We state the following theorem of our proposed algorithm.

Algorithm 3 Calculate$(v'_i,$ *updated_end*$)$ of c_i

Require: : v_i, $arr(c_i)$, $pst(c_i)$, *updated_end*
1: Get current time t_c
2: $\gamma = (arr(c_i) - t_c)/(pst(c_i) - t_c)$
3: **if** $\gamma >$ *thr*$_{up}$ and $\gamma <$ *thr*$_{down}$ **then**
4: $v'_i =$ *new_speed*(v_i, γ)
5: $arr' = v_i/v'_i \times (arr(c_i) - t_c) + t_c$
6: *updated_end* $= \max($*updated_end*$, arr')$
7: **else**
8: *updated_end* $= \max($*updated_end*$, arr(c_i))$
9: **return** : v'_i, *updated_end*

Theorem 1. *Let S be the schedule produced by the baseline scehdule-driven traffic control approach. The cumulative delay of schedule S', produced from schedule S by applying Algorithm 2, is less and equal to the cumulative delay of S*

Proof. We have two different situations. First, if the vehicle of c_i is speeding up, we know arr' is less than $arr(c_i)$. Although the delay contribution $|c_i| \cdot (ast(c_i) - arr') = |c_i| \cdot (arr' - arr') = 0$, we have smaller $ast(c_i) = \max(arr', pst(c_i)) = arr' < arr(c_i)$ and thus shorter phase duration pd. Second, if the vehicle of c_i is slowing down, arr' is larger than $arr(c_i)$ and $|c_i| \cdot (ast(c_i) - arr') \leq |c_i| \cdot (ast(c_i) - arr(c_i))$, where $ast(c_i) = pst(c_i)$. Thus, the cumulative delay is always less than the previous one. □

The cluster size $|c_i|$ affects the number of vehicles in each phase and thus has impact on how many vehicles will speed up to contribute to delay reduction in Algorithm 2. In the following section, we will discuss how clustering affects the performance of the proposed algorithm.

C. Cluster Size and Delay-Capacity Tradeoff

As mentioned in the previous sections, vehicles entering an intersection are clustered together if they are traveling within a pre-specified interval of one another. By changing this pre-specified interval, the cluster distribution is changed. If we have a larger interval, we could easily have large cluster size under high traffic demand. Although it could be computationally advantageous as the scheduling agent is computing schedule for large clusters and has higher throughput (or capacity, i.e., it does not require signal phases to switch too frequently), it may lose some benefits of using the cooperative algorithm, which requires better partition of platoons. On the other hand, it would be computational-intensive if the cluster size is always set to one. However, since a smaller interval is used, a favorable partition of platoon is generated with minimum cumulative delay, and it benefits the cooperative algorithm through either splitting

or merging clusters accurately. We can obtain a tradeoff between maximizing capacity and minimizing delay by both tuning the interval and applying the cooperative algorithm as well. The benefit of using the cooperative algorithm is different on the different intervals, but it is guaranteed that the delay of both large and small intervals is improved by Theorem 1.

Another possible way to improve the performance of applying a larger interval further is that we only reschedule the clusters with larger delay and their adjacent clusters in a single-vehicle scheduling manner, which is taking each vehicle as a single cluster. First, we have to identify a cluster c_i with the largest delay, which is $\Delta = (ast(c_i) - arr(c_i))$, and all the vehicles arriving during that interval as a single batch. Then, single-vehicle scheduling is applied, so that it is unnecessary to run single-vehicle scheduling on all vehicles within the current look-ahead horizon. Finally, we run the Algorithm 2 to change the speeds of vehicles within that horizon. To improve delay continually, we may identify a second cluster with larger delay and repeat above process. By incorporating this additional rescheduling, we can reshuffle vehicles in a batch to acquire smaller delay and more flexible computation compared to the original approach.

V. Experimental Evaluation

In this section, we evaluate the proposed cooperative algorithm through comparison to two different approaches: a baseline schedule-driven method and a fixed timing method in a connected vehicle environment with perfect information. The version of the schedule-driven method we apply here is Expressive Real-Time Intersection Scheduling (ERIS) [14], which maintains separate estimates of arrival traffic for each lane and enables it to more accurately estimate the effects of scheduling decisions than original schedule-driven approach. For accurate scheduling information, the proposed cooperative algorithm is also implemented based on ERIS. The fixed timing method is to match the given traffic demand according to Websters formula. We present a comparison over the proposed method, ERIS, and a fixed timing method on two separate networks.

The evaluations of the proposed method are ran on the Simulation of Urban MObility (SUMO), which is a microscopic traffic simulator that simulates continuous road traffic for large road networks and control of traffic signals and vehicles. To retrieve vehicle information and manipulate their behavior, we interface through Traffic Control Interface (TraCI) [15]. We assume that each vehicle has its own route as it passes through the network and measure how long a vehicle must wait for its turn to pass through the intersections (the delay or time loss). Tested traffic volume is averaged over sources at network boundaries. To assess the performance boost provided by the cooperative algorithm, we measure the average waiting time of all vehicles over five runs. All simulations run for 1 hour of simulated time. Results for a given experiment are averaged across all simulation runs with different random seeds. In order to eliminate the effects of simulation start up and termination, the time loss of vehicles arriving within the middle 40 minutes is reported. Vehicle arrivals are modeled as a Poisson process where the average arrival rate is set according to the desired level of congestion. We choose the IDM model with $\omega = 4$ for computing new speed, and other parameters are listed below: a) speed limit $65km/h$. b) *thr* $= 1 \pm 0.4$. c) $a_{max} = 5m/s$.

A. Simulation Results

First, a network consisting of a single intersection with two lanes on the main road and one lane on the side street is examined. To explore how the cooperative algorithm performs under different demand, we categorize traffic demand into three different groups: low (363 cars/hour), medium (750 cars/hour), and high (1250 cars/hour). Other than different traffic demand, we test how different clustering intervals affect the performance. Average delay per vehicle (in seconds) and standard errors across a range of vehicle volumes are presented in Table I(a). We also present the improvement when measured against fixed timing at each level of congestion.

With single-vehicle scheduling, where we have a smaller clustering threshold and take each vehicle as a single cluster, the cooperative algorithm outperforms schedule-driven approach and fixed timing plans for all tested levels of congestion. Under high traffic demand setting, the improvement of the cooperative algorithm is the largest and up to 19% and 33% compared to the two other approaches, which means more vehicles under this demand can be gained from the speed adjustment and scheduling. On the other hand, the performance is comparable for all three approaches under low and medium demand. The cooperative algorithm can only improve the delay of a small portion of vehicles, and most improvement in delay comes from obeying the produced schedule.

As the clustering threshold increases to 3 seconds, the improvement of the cooperative algorithm over schedule-driven traffic control becomes more evident and ranges up to 21% under high traffic demand. One reason is that for schedule-driven traffic control the reshuffling effect caused by scheduling is not able to create a platoon with smaller delay (37s v.s. 33s) because of large average cluster. However, the delay of the cooperative algorithm only increases from 27s to 29s. If sufficient computation power is available we would only suggest to use smaller clustering threshold in order to have better performance of the cooperative algorithm.

Other than single intersection, a model of three connected intersections is evaluated in Table I(b). The delay of the cooperative algorithm is still less than the schedule-driven traffic control approach under high traffic demand with two different clustering intervals as predicted in Theorem 1. However, the improvement is not as large as compared with single intersection. When we have three connected intersections, we observe that the delay with larger clustering interval is less than the case of single-vehicle scheduling as shown in the Table I(b). Basically, a smaller clustering interval provides a

	(a) Average Delay (second) of Single Intersection									
	Schedule-driven **(0s)**		Cooperative **(0s)**		Schedule-driven **(3s)**		Cooperative **(3s)**		Fixed timing	
	mean	std.	mean	std.	mean	std.	mean	std	mean	std.
High demand	33.52	26.86	27.23	21.86	37.52	37.84	29.67	26.83	40.62	28.87
Medium demand	19.32	20.10	17.75	15.41	19.62	17.07	17.83	15.67	25.67	18.64
Low demand	12.02	13.97	11.58	13.21	12.69	14.59	12.38	13.73	22.44	19.00
	(b) Average Delay (second) of Three Intersections									
	Schedule-driven **(0s)**		Cooperative **(0s)**		Schedule-driven **(3s)**		Cooperative **(3s)**		Fixed timing	
	mean	std.	mean	std.	mean	std.	mean	std	mean	std.
High demand	33.60	22.33	29.75	19.83	31.48	21.68	29.55	20.55	42.76	26.78
Medium demand	25.98	18.69	25.40	17.19	25.21	19.23	24.22	17.30	32.27	21.96
Low demand	17.82	15.93	18.10	15.27	12.69	14.59	12.38	13.73	28.19	19.71

TABLE I

AVERAGE DELAY OF SINGLE INTERSECTION AND THREE INTERSECTIONS WITH DIFFERENT CLUSTERING INTERVALS.

better optimality for single intersection, but it may cast more traffic demand for neighbor intersections and harm network-level performance (or global performance). A better network coordination should resolve this issue [16]. The delay of the cooperative algorithm in the second and forth column seems not be affected by the clustering interval in a network a lot. The shift of platoons somehow provides a robustness when applying traffic scheduling within a transportation network.

B. Partial Penetration of CAV

Understanding the performance under different penetration rates is an essential issue for deploying the proposed methods under realistic scenarios. In the results presented thus far, we have assumed that all vehicles are controllable by the proposed algorithm. However, it may be difficult in practice to achieve such a high penetration rate.

Table II presents the delay of the proposed algorithm under different penetration rates. We assume that partial vehicles are able to control their speed according to the messages sent by the intersection. Three penetration rates are tested to compare with full penetration rate. The results show that the proposed algorithm can still provide a considerable 14% improvement compared with the baseline case as the penetration rate is 50% and 70%. The improvement drops to 8% at a penetration rate of 30% which still represents a reasonable reduction in overall congestion.

	Average Delay (second)							
	100%		70%		50%		30%	
	mean	std.	mean	std.	mean	std.	mean	std.
High demand	27.23	21.86	29.22	23.64	29.57	24.31	31.58	26.89
Medium demand	17.75	15.41	19.12	17.50	19.29	16.38	19.40	17.28
Low demand	11.58	13.21	11.55	13.85	11.55	13.90	11.87	13.56

TABLE II

AVERAGE DELAY UNDER DIFFERENT PENETRATION RATES.

VI. CONCLUSION

In this paper, we described a cooperative framework designed to enable the interaction between CAV and schedule-driven traffic control. The scheduling algorithm generates useful scheduling information continually to reflect real traffic conditions by a strategy of frequent replanning. This information is used to guide vehicles on how to adjust their velocity for minimizing delay. The cooperative algorithm is specified as a way to shift approaching platoons through both reshuffling vehicles that are arriving from different, competing directions and changing their speed. The proposed approach demonstrates that a lower delay can be obtained. Moreover, we can still get reasonable performance under lower penetration rate of CAV.

The cooperative system was evaluated on a simulation model of a traffic signal control problem. Results showed that the interaction between scheduling and CAV improves average delay overall in comparison to both the baseline schedule-driven traffic control approach and a fixed timing approach, and that solutions provide substantial gain in highly congested scenarios. Future work will focus on the design of how to achieve a network-wide coordination through this vehicle-to-infrastructure communication.

REFERENCES

[1] X.-F. Xie, S. F. Smith, L. Lu, and G. J. Barlow, "Schedule-driven intersection control," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 168–189, 2012.

[2] S. F. Smith, G. J. Barlow, X.-F. Xie, and Z. B. Rubinstein, "Smart urban signal networks: Initial application of the surtrac adaptive traffic signal control system." in *ICAPS*. Citeseer, 2013.

[3] C. Priemer and B. Friedrich, "A decentralized adaptive traffic signal control using v2i communication data," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2009, pp. 1–6.

[4] N. Goodall, B. Smith, and B. Park, "Traffic signal control with connected vehicles," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2381, pp. 65–72, 2013.

[5] Q. He, K. L. Head, and J. Ding, "Pamscod: Platoon-based arterial multi-modal signal control with online data," *Transportation Research Part C: Emerging Technologies*, vol. 20, no. 1, pp. 164–184, 2012.

[6] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[7] R. Tachet, P. Santi, S. Sobolevsky, L. I. Reyes-Castro, E. Frazzoli, D. Helbing, and C. Ratti, "Revisiting street intersections using slot-based systems," *PloS one*, vol. 11, no. 3, p. e0149607, 2016.

[8] L. Li and F.-Y. Wang, "Cooperative driving at blind crossings using intervehicle communication," *IEEE Transactions on Vehicular technology*, vol. 55, no. 6, pp. 1712–1724, 2006.

[9] I. H. Zohdy and H. Rakha, "Game theory algorithm for intersection-based cooperative adaptive cruise control (cacc) systems," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 1097–1102.

[10] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.

[11] X. Liang, S. I. Guler, and V. V. Gayah, "Signal timing optimization with connected vehicle technology: Platooning to improve computational efficiency," *Transportation Research Record*, vol. 2672, no. 18, pp. 81–92, 2018.

[12] T.-C. Au and P. Stone, "Motion planning algorithms for autonomous intersection management," in *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[13] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[14] R. Goldstein and S. F. Smith, "Expressive real-time intersection scheduling," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*. ACM, 2008, pp. 155–163.

[16] H.-C. Hu and S. F. Smith, "Bi-directional information exchange in decentralized schedule-driven traffic control," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1962–1964.