Incorporating Queueing Dynamics into Schedule-Driven Traffic Control

Hsu-Chieh Hu[1] , **Allen Hawkes**[1] and **Stephen F. Smith**[1,2]
[1]Rapid Flow Technologies, Inc.
[2]Carnegie Mellon University Pittsburgh, PA, USA
{hsuchieh, allen}@rapidflowtech.com, sfs@cs.cmu.edu

Abstract

Key to the effectiveness of schedule-driven approaches to real-time traffic control is an ability to accurately predict when sensed vehicles will arrive at and pass through the intersection. Prior work in schedule-driven traffic control has assumed a static vehicle arrival model. However, this static predictive model ignores the fact that the queue count and the incurred delay should vary as different partial signal timing schedules (i.e., different possible futures) are explored during the online planning process. In this paper, we propose an alternative arrival time model that incorporates queueing dynamics into this forward search process for a signal timing schedule, to more accurately capture how the intersection's queues vary over time. As each search state is generated, an incremental queueing delay is dynamically projected for each vehicle. The resulting total queueing delay is then considered in addition to the cumulative delay caused by signal operations. We demonstrate the potential of this approach through microscopic traffic simulation of a real-world road network, showing a $10-15\%$ reduction in average wait times over the schedule-driven traffic signal control system in heavy traffic scenarios.

1 Introduction

As traffic congestion in cities continues to increase, it is generally recognized that better optimization of traffic signals is crucial to future urban mobility, and this fact, together with emergence of ubiquitous sensing capabilities, has led to new thinking around real-time approaches to this long studied problem (e.g., [Wongpiromsarn *et al.*, 2012; Varaiya, 2013; El-Tantawy *et al.*, 2013; Duncan *et al.*, 2014; Mitrovic *et al.*, 2019; Smith, 2020]. One promising approach that has emerged in recent years and is aimed specifically at urban traffic control, has been referred to as *schedule-driven traffic control*. [Xie *et al.*, 2012a; Xie *et al.*, 2012b; Smith *et al.*, 2013] This approach treats traffic control as a decentralized, online planning process, exploiting a novel formulation of the intersection control problem as a single-machine scheduling problem (where input jobs are sequences of spatially proximate vehicle clusters representing queues and approaching platoons). This aggregate representation enables generation of near-optimal timing plans for individual intersections in real-time, which are then shared with neighboring intersections to achieve network level coordination. Original work demonstrated substantial traffic flow efficiency improvements over statically optimized timing plans in actual field test experiments [Smith *et al.*, 2013]; and subsequent work has explored the further advantages of incorporating multi-modal traffic flows [Xie *et al.*, 2014], integrating with queue management objectives [Hu and Smith, 2017a; Hu and Smith, 2017b], exploiting higher fidelity predictive models [Goldstein and Smith, 2019], upstream propagation of congestion information [Hu and Smith, 2018; Hu and Smith, 2019], control of vehicle speed in connected vehicle setting [Hu *et al.*, 2019], use of stochastic sampling for modeling turning proportions [Dhamija *et al.*, 2020], and learning of model parameters [Hu and Smith, 2020]. Already incorporating many of these advances, this schedule-driven traffic control technology is now deployed and operating in 8 North American cities.

Despite this success, there is room for further improvement. One key to the effectiveness of any schedule-driven traffic control algorithm is an ability to accurately predict when the sensed vehicles will arrive at and pass through the intersection. All prior work in schedule-driven traffic control has assumed a static vehicle arrival model to make this prediction (as originally proposed in [Xie *et al.*, 2012a]). Initially, each vehicle cluster is associated with a free-flow arrival time, computed using a fixed free-flow speed estimate and the distance from vehicle sensor location to the intersection. In situations where queued vehicles are already present at the intersection, a delay proportional to the queue count (the number of the queued vehicles) is added to the free-flow arrival time, using the current queue count that remains unchanged, and this delay is propagated to all vehicle clusters within the planning horizon. The summation of the queueing delay and the free-flow arrival time to the end of queue is thus taken as a prediction of the estimated arrival time for each vehicle cluster.

One limitation of this static predictive model, however, is that it ignores the fact that the queue count and its incurred queueing delay should vary as different partial signal timing schedules (i.e., different possible futures) are explored during the online planning process. More specifically, the vehicle cluster's arrival time incorporating queueing delay depends

on when the previously queued vehicles are discharged and when it joins the queue, and this time can be better predicted through the previous partial schedule's state and its free-flow arrival time. We would expect to be able to further boost the performance of the traffic signal control system by introducing the ability to predict queueing delay online in response to observed traffic behavior.

In this paper, we propose an online queue prediction algorithm that incorporates queueing dynamics into this forward search process for a signal timing schedule. Our algorithm computes a more accurate estimate of the delay a queued vehicle will experience than static queue estimation techniques without increasing overall search complexity. As each search state is generated, an incremental queueing delay relative to the previous queued vehicle within the horizon is projected, based on the previous state, the free-flow arrival time, and the queue dissipation model. The resulting total queueing delay is considered in addition to the original cumulative delay caused by signal operations when evaluating alternative partial schedules. With this extension, we are able to better capture not only how the intersection's queues vary over time but also the impact of vehicle stops on cumulative delay at the intersection. We demonstrate the potential of this approach through microscopic traffic simulation of a real-world road network, showing a $10-15\%$ reduction in average wait times in heavy traffic scenarios over a schedule-driven variant that is utilizing the previous static predictive model.

2 Schedule-Driven Traffic Control

As noted above about the schedule-driven approach, the essential consequence of the single machine scheduling problem formulation of [Xie *et al.*, 2012b] is that traffic flows are treated 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 approaching an intersection from the entry roads are clustered together if the time gap between them is less than a pre-specified time interval (e.g., 1 second). The clusters then become the input jobs that must be sequenced through intersection (i.e., the single machine). Once a vehicle departs the intersection, it is sensed and grouped into a new cluster by the downstream neighbor intersection. The sequences of clusters provide short-term variability of traffic flows for optimization and preserve the non-uniform nature of real-time flows.

More specifically, the input to the online planning process at the beginning of each planning cycle is a set C of *phase cluster sequences*, where a *phase* is a compatible traffic movement pattern such as East-West traffic flow. A phase cluster sequence $C_{P,i}$ for a given phase i is obtained by merging those constituent *road cluster sequences* $C_{R,m}$ that can proceed concurrently through the intersection and belong to the phase, where each $C_{R,m}$ consists of $(|c|, arr, dep)$ triples that reflect each approaching or queued vehicle on entry road segment m and are ordered by increasing arr. The travel time on entry road m defines a finite horizon (H_m), and the prediction horizon H is the maximum over all road segments.

During planning, 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. In practice, the planning cycle is repeated every second or two, 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 m, and each time a phase change is implied by the sequence.

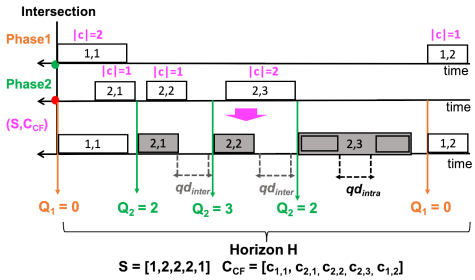


Figure 1: The resulting control flow (S, C_{CF}) and the predicted queue count Q at each search state: each block represents a vehicular cluster. The shaded blocks represent the delayed clusters.

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 $(c_1, \cdots, c_{|S|}), c_i \in C$ 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

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

where ast is the actual start time that the vehicle is allowed to pass through, which is determined by arr and permitted start time (pst). 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}, and the corresponding pst for c_k is equal to $t + MinSwitch(s, i)$, 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. The optimal sequence (schedule) C^*_{CF} is the one that incurs minimal delay for all vehicles.

3 Incorporating Queueing Dynamics into Scheduling

3.1 Queueing Delay

Other than the delay caused by the signal operations, i.e., $d(c) = |c| \cdot (ast - arr(c))$, a cluster may experience an extra delay, known as queueing delay, if the cluster's constituent vehicles are forced to decelerate or stop due to queued vehicles in front of them. In the static vehicle arrival model utilized in previous work, this delay is considered to be proportional to current *queue count* (the number of the queued vehicles in front of the approaching cluster). This count is assumed to

Algorithm 1 Calculate qd_{inter} and qd_{intra} of c

Require: 1) pst, a_0, a_1, s_k; 2) (s, pd, t, d) of S_{k-1}
1: $i = s_k$; c = next job of phase i; $qd_{inter}, qd_{intra} = 0$
2: $w = \max(0, |c| - 1)$ ▷ exclude the first vehicle of c
3: **if** $pst > arr(c)$ and c is not queued **then** ▷ c will stop
4: **if** $s \neq i$ and $arr(c) > t$ **then** ▷ New phase starts
5: $qd_{intra} = w \cdot (a_0 - a_1)$
6: $dep(c) = dep(c) + a_0 + \max(0, w - 1) \cdot a_1$
7: **else if** $arr(c) \leq t$ **then** ▷ Queue already exists
8: $qd_{inter} = a_1$
9: $dep(c) = dep(c) + w \cdot a_1$
10: **end if**
11: $qd_{intra} = qd_{intra} + (w + 1)w/2 \cdot a_1$
12: **end if**
13: **return** $(dep(c), qd_{inter}, qd_{intra})$

persist throughout the planning horizon and is not recomputed as different partial schedules are expanded.

To better capture the queueing dynamics in these situations, we propose to incorporate an alternative vehicle arrival model where queueing delay evolves as a function of different partial signal timing schedules (i.e., different possible futures) that are explored during the online planning process. More specifically, as the search states representing different partial schedules are expanded to incorporate the next approaching cluster, the possibility of queuing delay is assessed. If the arr of the next cluster to be added to the schedule, computed using a fixed free-flow speed estimate and the distance from vehicle sensor location to the intersection, is less than the finish time t of the previous search state (i.e., the intersection departure time of the previous cluster), then this cluster is determined to have joined the queue. In this case, the *headway* (i.e., the temporal gap between two adjacent queued vehicles) of all preceding vehicles in the queue ahead of the cluster is summed and added to the cluster's free-flow arr as the queueing delay, and arr and dep of the vehicle clusters will be thus delayed.

To incorporate the queueing delay into the objective, a queueing delay that is relative to ast is added to the original cluster delay, and the total delay contributed by cluster c is rewritten as

$$d(c) = |c| \cdot (ast - arr(c)) + qd(c), \quad (2)$$

where $qd(c)$ is the queueing delay of c and can be de determined through the search process and a queue dissipation model that will be introduced in the next section.

3.2 Queue Dissipation Model

The total queueing delay for a single vehicle due to the existing standing queue can be estimated using the following relationship adopted from [Head, 1995]):

$$qd_{total}(N_q) = a_0 + a_1 \cdot (N_q - 1), \quad a_0 \geq a_1; N_q \geq 1 \quad (3)$$

where a_0 and a_1 are parameters that can be selected on the basis of the particular intersection and N_q is the number of vehicles already in the queue. The interpretation of this model is that a_0 is a sum of the initial headway and residual start-up lost time for the second vehicle in the queue, and the following vehicles joining the queue experience an additional headway a_1. Since the initial speed is small, a_0 should be greater than a_1. With this model, we can estimate how much delay a queued vehicle will experience before being dissipated and the headway relative to the previous queued vehicle. Although the model can be extended to a more complicated polynomial or non-linear functional form, our experimental data shows that a linear model is sufficient to model dissipation. The parameters can be selected by field experiments or learned by online learning algorithms.

To retain the efficiency of generating longer-horizon plans, we define two types of queueing delay for a vehicle cluster that contains multiple vehicles: 1) *intra-cluster queueing delay* and 2) *inter-cluster queueing delay*. *Intra-cluster queueing delay* is the delay experienced by all vehicles other than the first vehicles within the cluster, and it can cause the enlargement of the cluster (i.e., $dep(c)$ increases). *Inter-cluster queueing delay*, alternatively, is the delay experienced by the first vehicle within the cluster if there are already one or more queued clusters in front of it, and the pst will be delayed. For the example of Figure 1, the stop of cluster $(2, 1)$ causes a qd_{inter} between $(2, 1)$ and $(2, 2)$, and the stop of $(2, 2)$ causes qd_{inter} between $(2, 2)$ and $(2, 3)$ and qd_{intra} within $(2, 3)$. With these definitions, Equation 2 can be rewritten as

$$d(c) = |c| \cdot (ast + qd_{inter} - arr(c)) + qd_{intra}, \quad (4)$$

where qd_{inter} and qd_{intra} are the inter-cluster and intra-cluster queueing delay and $qd(c) = |c| \cdot qd_{inter} + qd_{intra}$. qd_{intra} and qd_{inter} can be calculated given a_0, a_1 and the state tuple (s, pd, t, d) by Algorithm 1.

Algorithm 2 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: Compute qd, and update $dep(c)$ and pst:
a: $pst = t + MinSwitch(s, i)$;
b: $(dep(c), qd_{inter}, qd_{intra})$ =Algorithm 1, given (pst, a_0, a_1), input 1) and 2); ▷ Queueing dynamics
c: $pst = pst + qd_{inter}$; ▷ insert qd_{inter} if needed
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: **end if**
6: $t = ast + dep(c) - arr(c)$ ▷ Actual finish time of c
7: **if** $s \neq i$ **then** $pd = t - pst$
8: **else** $pd = pd + (t - pst)$
9: **end if**
10: $d = d + |c| \cdot (ast - arr(c)) + qd_{intra}$ ▷ Total delay
11: **return** (pd, t, d)

In Algorithm 2, we describe how the search state S_k is expanded from S_{k-1}, given the previous tuple (s, pd, t, d), based on a greedy realization of planned signal sequence. Algorithm 1 is executed at Line 2 of Algorithm 2, and then qd_{inter}, qd_{intra}, and $dep(c)$ are derived for revising pst and delay contribution accordingly. Then, ast is determined by the maximum of arr and pst at Line 3 of Algorithm 2. If arr is less than pst, this means that the cluster will stop and the both types of queueing delay should be included. We consider

two cases after knowing the cluster will stop. First, if a cluster joins an existing queue, then its qd_{intra} is calculated based on the following theorem:

Theorem 1. *If a cluster c joins an existing queue, the incurred queueing delay within the cluster is $O(|c|^2)$ and can be calculated by $qd_{intra} = \frac{(|c|-1)|c|}{2} \cdot a_1$*

Proof. The queueing delay experienced by the second vehicle within the cluster is a_1; The one experienced by the third vehicle is $2a_1$ contributed by the queueing delay from the previous vehicle and itself, and so on. qd_{intra} experienced by the vehicles within c other than the first vehicle is thus $a_1 + \cdots + (|c|-1) \cdot a_1 = \frac{(|c|-1)|c|}{2} \cdot a_1$. □

Then, qd_{inter} is set to a_1 and added to pst when the cluster joins an existing queue. Its contribution to the cumulative delay is $|c| \cdot a_1$. Equation 4 can be rewritten as

$$d(c) = |c| \cdot \big(ast + a_1 - arr(c)\big) + \frac{(|c|-1)|c|}{2} \cdot a_1 \quad (5)$$

Second, if a new phase has just started in the currently expanded search state, then the second vehicle within the added cluster will experience headway a_0 instead of a_1 and the rest of vehicles will still experience a_1 instead. Therefore, the queueing delay needs to be offset by the difference $(a_0 - a_1)$. Note that there is no qd_{inter} in this case since it is the first cluster in the new phase.

3.3 Online Planning with Queueing Delay

In this section, we describe how to integrate queueing delay into the online planning algorithm and predict the queue count. The goal is to improve traffic control performance by utilizing a more accurate predictive queueing model.

First, before computing the schedule, we determine whether a vehicle is queued or not given the snapshot of current queue count. Then, all vehicles approaching the intersection are clustered together based on the free-flow arr at their free-flow speed. In other words, approaching vehicles are clustered according to their geometrical proximity. After clustering, we delay arr of the queued clusters based on the snapshot by adding Equation 3 according to the queue count they observe when joining the queue, and this delayed arr will serve as a part of the initial state for building the schedule.

When computing the schedule, each cluster will be checked if it joins a queue. If so, $dep(c)$ of the free-flow clusters will be increased, and the queueing delay is computed to update pst and the objective as described in Algorithm 2 and 1 at each expansion step of the search.The physical interpretation of this update is that the stop due to the current queue expands the cluster duration dur by $a_1 \cdot \max(0, |c|-1)$ and increases the gap between clusters' arrival time by qd_{inter}.

Since the previously queued clusters will depart the intersection at the $ast = \max(pst, arr(c))$ that is determined by the schedule, we can predict the queue count at any future time points within the horizon H by tracking the number of the dissipated vehicles and the number of vehicles joining the queue at each search state. Three cases mentioned in Algorithm 1 are necessarily considered for predicting the queue count: a) If the cluster does not stop, the queue count of this cluster is 0. b) If the cluster stops and a queue does not exist (i.e., a start of a new phase), the queue count is 0 as well. c) If the cluster stops and a queue already exists, given the queue count of the previous state, then the queue count of the current state is represented as

$$Q(S_k) = Q(S_{k-1}) + |c_{k-1}| - l(S_k), \quad (6)$$

where $Q(S_k)$ is the observed queue count of the partial schedule S_k for c_k, and $l(S_k)$ is the number of dissipated vehicles between the arrivals of c_k and c_{k-1}.

More specifically, the number of the vehicles joining the queue at each state is simply $|c_{k-1}|$ of the previous cluster in the same road cluster sequence $C_{R,m}$ on road segment m. However, knowing the number of the dissipated vehicles requires to count how many vehicles whose end times t fall within the range between two adjacent arrivals are dissipated. An algorithm that computes $Q(S_k)$ in the case c) is provided in Algorithm 3,

Algorithm 3 Predict $Q(S_k)$ given S_{k-1} and c_k

Require: 1) S_{k-1},$Q(S_{k-1})$; 2) c_{k-1} and c_k
1: $l(S_k) = 0$;$ptr = c_{k-1}$
2: **while** $\max(\textit{phase start}, arr(c_{k-1})) < ast(ptr)$ **do**
3: **if** $t(ptr) < arr(c_k)$ **then** ▷ check end time t of ptr
4: $l(S_k) = l(S_k) + |ptr|$ ▷ ptr is dissipated
5: **end if**
6: $ptr =$ previous cluster of the cluster ptr
7: **end while**
8: $Q(S_k) = Q(S_{k-1}) + |c_{k-1}| - l(S_k)$

Since the extension does not generate additional states during the search, it retains the efficiency of scheduling with longer horizons, and the complexity is still *polynomial* in $|H| = H/\delta$, the number of time steps in H [Xie *et al.*, 2012b], given a time resolution δ.

Currently, the baseline arr for calculating delay is assumed to be arrival time of the vehicle cluster if moving according to the static arrival model. In the following section, a concept of minimum guaranteed queue is introduced to define the objective (cumulative delay) for taking both queueing delay and signal delay into consideration.

3.4 Minimum Guaranteed Queue

According to Equation 1, each cluster contributes a time difference between the ast and the free-flow arr to the cumulative delay. After taking queueing delay into account, ast is offset by a queueing delay (equally a headway), i.e, qd_{inter}, and qd_{intra} are added according to Equation 4. In turn, the delay contribution of each cluster should include the total queueing delay caused by all previous queued vehicles other than the signal delay. However, to define the cluster delay correctly, arr should be a delayed arr in which we assume each cluster will experience its own minimum queue count if the corresponding green phase persists.

Definition 1 (Minimum Guaranteed Queue)**.** *Minimum Guaranteed Queue (MGQ) is the minimum queue count a moving cluster observes when it joins the queue under an assumption that the corresponding phase stays green until the cluster exits.*

MGQ is used to specify arr of the cluster delay (i.e., Equation 5) when both qd_{intra} and qd_{inter} are considered and taken as a baseline to define delay. Meanwhile, the minimum queueing delay without any intervention of the traffic signal can be calculated directly from MGQ. However, since MGQ is irrelevant to any state transition, the solutions will only differ by a constant if another baseline arr, e.g., arr at the free-flow speed, is applied.

Theorem 2. *The solutions of any baseline arr will only differ by a constant compared to the arr calculated with MGQ.*

Proof. arr is not state-dependent due to the traffic signal operations. Therefore, we can replace it with any baseline arr without compromising the optimality of the solution. □

Due to Theorem 2, we simply use the free-flow arr to compute the schedule without loss of optimality unless there is a need for acquiring the actual cost of the schedule.

4 Experimental Evaluation

In this section, we compare the above described online queue prediction algorithm to three other real-time traffic control methods. First, we include the two most recent variants of the schedule-driven traffic control system: a) The variant that ensures queueing stability (i.e., that queues will not increase without bound) [Hu and Smith, 2017a; Hu and Smith, 2017b] serves as the primary benchmark since this version has been proved to be effective in several North America cities [Smith, 2020], and b) the more recent bi-directional extension that utilizes the downstream congestion information [Hu and Smith, 2018; Hu and Smith, 2019] is also included as another point of comparison. We implement new versions of each of these two extensions that incorporate online queue prediction. Second, we compare to a variant of backpressure adaptive control [Wongpiromsarn *et al.*, 2012; Varaiya, 2013] that makes decisions based on the estimated queue count and ensures queueing stability. [1]

To evaluate our approach, we simulate performance on a two-intersection model and a real world network. The two-intersection model is used to estimate model parameters. The real world network is used to evaluate the performance of planning with queueing delay in a larger complex real network and real traffic pattern. The simulation model was developed in VISSIM, a commercial microscopic traffic simulation software package. 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). Tested traffic volume is averaged over sources at network boundaries. To assess the performance boost provided by the proposed 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.

[1]Note also that previous research with the baseline schedule-driven approach has shown its comparative advantage over prior online planning approaches to real-time traffic signal control [Xie *et al.*, 2012a].

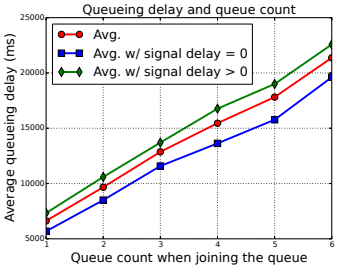


Figure 2: The average queueing delay for different queue counts.

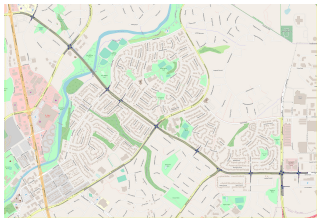


Figure 3: Map of the 11 intersections in the St. Albert, Canada

Estimation of Model Parameters Since the headway parameters a_0 and a_1 are critical for computing the queueing delay, setting them correctly could affect how the queueing dynamics are modeled. We consider a simple two-intersection model with 2-way, single lane, and multi-directional traffic flow as controlled experiments. The source of traffic is assumed to be stationary and set to 500 cars/hour. In Figure 2, three scenarios are plotted. We can see that the three lines actually fit the queue dissipation model (i.e., Equation 3) closely, which can be characterized by an intercept (i.e., a_0) and a slope (i.e., a_1). The difference between them is in whether vehicles experience a signal delay or not. If the signal delay is larger than 0, an additional start-up lost time should be added to the average delay, leading to an upper shift of the line. We use a linear function to approximate the line without the signal delay, and the estimations of a_0 and a_1 are approximately $(5.8s, 2.4s)$. We then apply these values to a larger complex network. For non-stationary traffic, a_0 and a_1 can be adjusted adaptively through tracking arrival and departure of vehicles. It is worth noting that the performance of this model shows the same trend as the following urban network model.

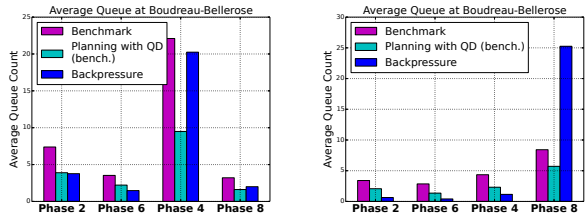
Urban Network Model The network model we consider for a more complex scenario is based on the St. Albert neighborhood of Canada as shown in Figure 3. The network consists of 11 intersections that basically have multiple phases. It can be seen as a two-way corridor network. To explore how the proposed algorithm performs under different traffic patterns and demands, we evaluate two traffic patterns: AM and PM rush hour, and categorize traffic demand into three different groups: low (AM/PM $116.18/66.55$ cars/hour), medium (AM/PM $249.27/99.82$ cars/hour), and high (AM/PM $332.36/133.10$ cars/hour). The low demand data is extracted from the field data of St. Albert of 6-9am, 1/6/2020 - 1/8/2020, and ramped up to generate two other demands.

Table 1 shows that both versions of the proposed method are basically comparable except that the one combining with benchmark (i.e., ensure queueing stability) performs better than the other by 2.4% under the AM high demand case. In the following results, we mainly consider this variant to present our evaluation. It shows the online queue prediction to yield a delay improvement over the benchmark and the bi-directional extension of about $13\%/19\%$ and an improvement of about 20% over backpressure control for the AM high traffic demand case. For the number of stops, the improvements are 24%, 23% and 48% respectively. For low and medium traffic, the de-

	Average Delay (second) and Number of Stops																			
	Benchmark				Bi-directional				Planning with qd (benchmark)				Planning with qd (bi-direc.)				Backpressure			
	mean	std.	stop no.	std.	mean	std.	stop no.	std.	mean	std.	stop no.	std.	mean	std.	stop no.	std	mean	std.	stop no.	std
AM High	95.55	95.87	3.89	7.34	104.57	103.94	3.66	7.19	**84.77**	**81.10**	**2.96**	**4.33**	86.83	82.12	3.03	4.53	105.61	118.63	5.68	10.94
AM Medium	64.86	60.98	1.74	1.80	62.96	57.23	1.65	1.62	**56.04**	**52.94**	**1.60**	**1.50**	56.64	53.31	1.61	1.54	59.29	55.39	1.73	1.67
AM Low	54.11	53.68	1.40	1.25	53.59	52.87	1.44	1.3	**49.54**	**49.41**	**1.37**	**1.22**	49.37	49.57	1.38	1.25	52.16	51.66	1.39	1.24
PM High	50.08	50.36	1.29	1.11	49.57	49.73	1.33	1.21	**47.75**	**48.78**	**1.29**	**1.13**	48.91	49.12	1.32	1.15	51.97	53.5	1.33	1.15
PM Medium	47.20	49.67	1.25	1.10	46.54	49.16	1.23	1.19	**45.73**	**47.31**	**1.22**	**1.07**	46.22	47.55	1.23	1.09	51.02	53.63	1.25	1.07
PM Low	44.58	48.26	1.20	1.01	43.85	48.32	1.19	1.01	**46.11**	**48.80**	**1.20**	**1.02**	45.78	48.60	1.19	1.18	55.20	58.03	1.21	1.02

Table 1: The mean and standard deviation (std.) of delay and number of stops (stop no.) under different scenarios.

lay and the stops of the proposed approach are generally better than the three approaches. The online queue prediction enabling planning with the queueing delay prevents the decision making from falling into short-sighted decisions and allows it to outperform backpressure, which only utilizes current queue counts to make decisions. On the other hand, compared to the benchmark, which is actually comparable to the bi-directional extension, taking the queueing delay into account more accurately captures realistic traffic variation and improves the schedule. Moreover, from the AM high demand case, we can see that maintaining queueing stability and using prediction are both crucial for reducing congestion.



(a) Boudreau-Bellerose (b) Boudreau-Campbell

Figure 4: Average queue counts at 2 most congested intersections in the network

For the PM rush hour, although the demand is not as large as the AM rush hour, several dominating roads are present instead of a single main road like the AM rush, requiring better coordination among the intersections. Because of the lower demand, the backpressure approach is not able to reduce delay through the simple queueing policy. Contrarily, the backpressure's weakness to coordinate intersections is to worsen the delay of entire network, so it has the largest delay among four approaches under the PM low demand. The proposed online queue prediction is generally better than the benchmark and the bi-directional extension, even in PM rush hour it has shorter queues. The benefits come from the fact that the proposed vehicle arrival model is more accurate at predicting arrival times.

Figure 4 also shows the average queue counts of two bottleneck intersections, Boudreau-Bellerose and Boudreau-Campbell, under the AM high demand scenario. The average queue count of four roads at both intersections are shown. The proposed approach has the shortest queue, compared to the benchmark and the backpressure approach respectively. As can be seen, the use of online queue prediction reduces the queue count and balances the queues among four roads through planning. For example, compared to the backpressure, although the proposed approach has longer queues at the first three roads of Boudreau-Bellerose, the reduction of the queue count on the forth road is significant in return. It is the same for the third road at Boudreau-Campbell.

Table 2: RMSE and lookahead h of using 2nd and 3rd cluster to predict queue count at Boudreau-SWC and their corresponding average queue count

	2nd Cluster				3rd Cluster			
	Avg.	[±3s]	[±5s]	h(second)	Avg.	[±3s]	[±5s]	h(second)
Phase 2	1.92	1.22	1.11	26.91	1.80	1.24	1.09	41.16
Phase 8	1.53	0.73	0.63	20.92	1.94	0.94	0.86	29.53

According to Algorithm 3, future queue count is predicted during generating schedule and tracking arrivals/departures of the clusters. In Table 2, two comparisons on different phases (i.e., 2 and 8) at Boudreau-Sir Winston Churchill (SWC) intersection are made. We take the second and the third cluster (i.e., c_2 and c_3) as predictions of the future queue count. Due to the uncertainty of arrival time, the measured queue count that is within a window (e.g., ± 3 or ± 5 second) and has the smallest error is chosen to compute root-mean-square error (RMSE). The prediction error is around 1 vehicle, and the average lookahead h(second) could be up to 40 second.

5 Conclusion

In this work, we considered the limitations of prior approaches to schedule-driven traffic control that rely on a static arrival model without regard to the fact that the queue count and the incurred delay should vary as different partial signal timing schedules are explored during the online planning process. An online queue prediction algorithm is proposed to achieve better delay and number of stops in circumstances of high traffic demand. In this algorithm, each scheduling agent computes queueing delay of each cluster dynamically, as states are expanded during the search process, given the previous state and the free-flow arrival time, and predicts the arrival time dynamically. Experimental results showed that the proposed approach improves cumulative delay overall in comparison to the schedule-driven traffic control approaches using a static vehicle arrival model and a backpressure approach, and that solutions provide substantial gain in highly congested scenarios.

Acknowledgements

Stephen F. Smith's work was supported in part by the National Science Foundation, under award #2038612, and the Mobility21 University Transportation Center at CMU.

References

[Dhamija *et al.*, 2020] Srishti Dhamija, Alolika Gon, Pradeep Varakantham, and William Yeoh. Online traffic signal control through sample-based constrained optimization. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 366–374, 2020.

[Duncan *et al.*, 2014] Gary Duncan, K Larry Head, and R Puvvala. Multi-modal intelligent traffic signal system-safer and more efficient intersections through a connected vehicle environment. *IMSA Journal*, 52(5), 2014.

[El-Tantawy *et al.*, 2013] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1140–1150, 2013.

[Goldstein and Smith, 2019] Rick Goldstein and Stephen F Smith. Expressive real-time intersection scheduling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9882–9883, 2019.

[Head, 1995] Larry K Head. Event-based short-term traffic flow prediction model. *Transportation Research Record*, 1510:45–52, 1995.

[Hu and Smith, 2017a] Hsu-Chieh Hu and Stephen F Smith. Coping with large traffic volumes in schedule-driven traffic signal control. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 154–162, 2017.

[Hu and Smith, 2017b] Hsu-Chieh Hu and Stephen F Smith. Softpressure: a schedule-driven backpressure algorithm for coping with network congestion. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4324–4330, 2017.

[Hu and Smith, 2018] Hsu-Chieh Hu and Stephen F Smith. Bi-directional information exchange in decentralized schedule-driven traffic control. In *Proceedings of the international conference on autonomous agents and multiagent systems*, pages 1962–1964, 2018.

[Hu and Smith, 2019] Hsu-Chieh Hu and Stephen F. Smith. Using bi-directional information exchange to improve decentralized schedule-driven traffic control. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 200–208, 2019.

[Hu and Smith, 2020] Hsu-Chieh Hu and Stephen F Smith. Learning model parameters for decentralized schedule-driven traffic control. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 531–539, 2020.

[Hu *et al.*, 2019] Hsu-Chieh Hu, Stephen F Smith, and Rick Goldstein. Cooperative schedule-driven intersection control with connected and autonomous vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1668–1673. IEEE, 2019.

[Mitrovic *et al.*, 2019] Nikola Mitrovic, Igor Dakic, and Aleksandar Stevanovic. Combined alternate-direction lane assignment and reservation-based intersection control. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1779–1789, 2019.

[Smith *et al.*, 2013] Stephen F Smith, Gregory J Barlow, Xiao-Feng Xie, and Zachary B Rubinstein. Smart urban signal networks: Initial application of the surtrac adaptive traffic signal control system. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 434–442, 2013.

[Smith, 2020] Stephen Smith. Smart infrastructure for future urban mobility. *AI Magazine*, 41(1):5–18, 2020.

[Varaiya, 2013] Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.

[Wongpiromsarn *et al.*, 2012] Tichakorn Wongpiromsarn, Tawit Uthaicharoenpong, Yu Wang, Emilio Frazzoli, and Danwei Wang. Distributed traffic signal control for maximum network throughput. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 588–595. IEEE, 2012.

[Xie *et al.*, 2012a] Xiao-Feng Xie, Stephen F Smith, and Gregory J Barlow. Schedule-driven coordination for real-time traffic network control. In *ICAPS*, 2012.

[Xie *et al.*, 2012b] Xiao-Feng Xie, Stephen F Smith, Liang Lu, and Gregory J Barlow. Schedule-driven intersection control. *Transportation Research Part C: Emerging Technologies*, 24:168–189, 2012.

[Xie *et al.*, 2014] X. Xie, S. F. Smith, T. Chen, and G. J. Barlow. Real-time traffic control for sustainable urban living. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1863–1868, 2014.