Platoon-Based Self-Scheduling for Real-Time Traffic Signal Control

Xiao-Feng Xie, Gregory J. Barlow, Stephen F. Smith, and Zachary B. Rubinstein

***Abstract*— In this paper, we take a self-scheduling approach to solving the traffic signal control problem, where each intersection is controlled by a self-interested agent operating with a limited (fixed horizon) view of incoming traffic. Central to the approach is a representation that aggregates incoming vehicles into critical clusters, based on the non-uniformly distributed nature of road traffic flows. Starting from a recently developed signal timing strategy based on clearing *anticipated queues*, we propose extended real-time decision policies that also incorporate look-ahead of approaching vehicle *platoons*, and thus focus attention more on keeping vehicles moving than on simply clearing queues. We present simulation results that demonstrate the benefit of our approach over simple queue clearing, both in promoting the establishment of "green waves" where vehicles move through the road network without stopping and in improving overall traffic flows.**

I. Introduction

Traffic signal control is an important practical problem. A recent study on urban mobility in 439 urban areas of the USA [1] indicates that the delay and fuel cost due to traffic congestion was $115 billion in 2009. Moreover, the congestion has negative impacts on environmental conditions. It is generally recognized that improved traffic signal control offers the biggest payoff for reducing congestion on surface streets, and that signal control systems that adjust their settings to fit current traffic conditions (as opposed to more conventional, *fixed* signal timing systems) offer the most potential. Although there are examples of such *adaptive* traffic signal control systems in practice [2]–[10], there continues to be a recognized gap in technologies that can respond efficiently and effectively to real-time changes in traffic demand. The problem is quite challenging. On one hand, the number of joint timing plans and traffic conditions is huge for even an individual intersection [11], [12], and grows exponentially with the size of the traffic network [13]. On the other, the non-linear dynamics of a switched network [14] and unpredictable human driving behaviors make reliable prediction possible only over a limited time horizon and forces continual change to computed solutions.

Given the problem's complexity and dynamics, we adopt a self-scheduling framework. Self-scheduling systems have gained increasing attention in recent years [15], especially as a means for managing the execution of complex processes in dynamic environments. Such systems are composed of a collection of autonomous agents, each with responsibility for controlling some portion of the overall process. The coordination between agents is normally achieved implicitly by exploiting structural information in their shared environment, although explicit coordination is not precluded.

We assume each intersection is controlled by a distinct self-interested agent, operating with a limited (fixed horizon) view of incoming traffic (as provided by the sensors on incoming roads [16]). Central to our approach is an aggregate representation of traffic flows as critical clusters of *anticipated queues* and *platoons*. These aggregate patterns provide a basis for real-time signal control policies that incorporate greater look-ahead and extend the queue clearing strategies utilized by prior work (notably [8]). We design two platoon-based policies aimed at deciding whether or not to extend a phase through idle periods of no traffic in order to service future traffic, with the goal of promoting implicit coordination between successive signals and the establishment of "green waves" [17], and ultimately improving overall performance. We present simulation results on two traffic networks with dynamic vehicle flows that demonstrate the performance benefit of our self-scheduling approach.

II. Related Work

Classical traffic control strategies assume a cyclic operation of traffic lights, where each light moves through its sequence of phases in a cycle that is offset from those of its neighbors. Explicit coordination then tries to facilitate "green waves" [17] over a set of successive intersections. In most conventional traffic control systems, cycle timing plans are built offline by choosing the *cycle lengths*, *phase split times*, and/or *offsets* for lights, and in some adaptive approaches, these parameters are adjusted over time [2], [3], [11]. However, these methods typically require some degree of stability in traffic flows over time, to enable coordinating traffic lights to build knowledge of macroscopic patterns; they generally cannot exploit microscopic (e.g., second-by-second) variability in prevailing flows [4].

Coordination between traffic lights can also be achieved implicitly through joint perception of the shared environment in which they operate. Implicit coordination has been achieved to some extent in model-based optimization strategies [4], [5], [11] by exploring possible states, but computation is expensive if the planning horizon is long, and it is difficult for such "black-box" optimization approaches to take advantage of flow pattern information. Another implicit approach to coordination is self-organization, which tries to use simple clues of global patterns in the distributed environment [6]–[9]. In [6], a self-organizing traffic light uses a threshold to control a time-based integration of the count of arriving vehicles within a given horizon during the red light, so that large enough "platoons" can move without

The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 {xfxie,gjb,sfs,zbr}@cs.cmu.edu

unnecessary stopping. More recently, a self-control method was proposed based on the notion of an "anticipated queue", which contains not only those vehicles already queued at the intersection, but also those predicted to reach the intersection before the last queued vehicle is cleared [8]. Our method incorporates this idea, but the focus shifts from minimizing queues [7], [8] to maintaining movement of vehicle platoons.

III. Problem Formulation

For each intersection in a traffic network, there are a set of roads from which vehicles enter the intersection and a set of roads from which vehicles depart the intersection. The traffic light at any given intersection cycles through a fixed sequence of phases, and each phase governs right of way for a set of non-conflicting movements through the intersection. Additionally, there are safety and fairness constraints. The yellow light between any two phases runs for a fixed duration (Y), while the green period for each phase has a variable duration between a minimum (G_{min}) and maximum (G_{max}). The objective of the intersection controller is to allocate green time to phases so as to minimize the total delay of vehicles traveling through a road network.

For simplicity, we assume that each intersection operates with two phases, each of which services a one-way road. For a more complex intersection, the movements in each phase can be merged by using estimated turning proportions [4].

IV. Platoon-Based Self-Scheduling

Our proposed approach to intersection signal control, referred to as platoon-based self-scheduling (PBSS), proceeds according to a rolling horizon [4], [5]. At each decision point, the intersection signal controller has two possible actions [5]: (1) to extend the current phase of the light for a duration $t_{ext} > 0$, or (2) to terminate the current phase and switch to the next phase ($t_{ext} = 0$). The extension time t_{ext} is capped by $(G_{max} - t_{ge})$, where t_{ge} is the current elapsed green time in the current phase. If the current phase is terminated, two signal timings are installed: the yellow time Y followed by the minimum green duration G_{min} for the next phase.

The input information available at a given decision point includes the current phase of the traffic light, t_{ge}, and the primary flow data sensed on all incoming roads.

A. Primary Flow Data

For each incoming road, the flow data contain two elements (qn, C), in which qn is a count of the vehicles in the queue at the intersection and C is an ordered sequence of clusters in a prediction horizon [12]. Each *cluster* c can be represented as a tuple $(|c|, arr_c, dep_c)$, where $|c|$ is the number of vehicles in the cluster, and arr_c (dep_c) gives the expected arrival time (departure time) respectively for the first (last) vehicle in c. For a cluster c, two associated values can be calculated: the duration ($dur_c = dep_c - arr_c$) and the flow rate when the cluster is serviced ($flow_c = |c|/dur_c$).

The data are collected by using a simple technique [16]. On each incoming road to an intersection, there are two sensors, one at the stop line at the intersection and the other at a fixed distance (L_{det}) upstream on the road. On each road, each vehicle travels at a constant *flow speed* v_F [4]. The advance detector defines the prediction horizon [12], i.e., L_{det}/v_F. Incoming vehicles are tracked by periodically sampling the upstream sensor. Each sampling period can potentially introduce a new cluster into C, and a given cluster remains in C until the corresponding vehicles arrive at the intersection. The vehicles crossing through the intersection are monitored via the stop-line detector. The queue size qn is the difference between arrived and departed vehicles.

B. Aggregation

The aggregation process is used to organize non-uniformly distributed traffic flow into simple patterns. Two simple techniques are exploited. After summarizing each, some basic intuition behind the aggregate representation is discussed.

1) Clustering: A simple threshold-based clustering is used to aggregate the sampled clusters in C into bigger ones. Two primitive clusters are merged whenever they are found to be within a specified temporal distance (th_g) of each other. If c is a merged cluster, then arr_c is the minimum arr of the constituent clusters merged, dep_c is the maximum dep, and $|c|$ is the sum of the constituent cluster counts.

If $th_g = 0$, C is unchanged. If th_g is large enough, then all clusters are merged into one cluster of uniformly distributed vehicles, similar to the simplification in the store-and-forward model [11].

2) Anticipated Queue: For a given incoming road, the flow data (qn, C), can be used to anticipate the number of vehicles qa that are either presently in the queue at the intersection or will join it before it clears [8].

The clearance time for a queue of size qa is a generalization based on a simple model [16],

$$QCT(qa, t_o) = \max(t_{SL} + t_o, 0) + qa/flow_{SAT} \qquad (1)$$

where t_{SL} is the start-up lost time, $flow_{SAT}$ is the saturation flow rate, and t_o is an offset time related to the traffic light. $t_o = -t_{ge}$ for the current green phase, and $t_o \geq 0$ designates the amount of time until the light switches to green in the case of a future phase. If $t_o = 0$ then QCT returns the actual clearance time. Based on Eq. 1, a queue can be seen as a cluster that has the saturation flow rate $flow_{SAT}$ and arrives after the time $\max(t_{SL} - t_o, 0)$.

The anticipated number of vehicles in the queue (qa) is calculated by using Algorithm 1. Initially, $qa = qn$ (Line 1), but some clusters in C might join the queue before the qn vehicles in the queue are cleared. Because the clusters in C are ordered, the calculation is started from the earliest cluster c_i in C (Line 2). There are several cases for extending the anticipated queue. The cases in Line 6 are straightforward, and all vehicles in the cluster c_i will join qa. In Line 9, Δdur is the duration of the portion of the ith cluster that will join the anticipated queue.

There are two differences in Algorithm 1 as compared to the calculation in [8]. First, it is based on a discrete and non-uniform distribution of the incoming traffic flow instead of a continuous function of the vehicle arrival rate, which is

Algorithm 1 Estimate qa, given t_o and (qn, C)

1: $qa = qn$
2: **for** $i = 1$ to $|C|$ **do**
3: $t = QCT(qa, t_o)$ {using Eq. 1}
4: **if** $arr_{c_i} \leq t$ **then**
5: $\Delta flow = flow_{SAT} - flow_{c_i}$
6: **if** $\Delta flow \leq 0$ or $dep_{c_i} \leq t$ **then**
7: qa = qa+$|c_i|$ {all vehicles in c_i join in qa}
8: **else**
9: $\Delta dur = (t - arr_{c_i}) \cdot flow_{SAT}/\Delta flow$
10: **if** $\Delta dur \geq dur_{c_i}$ **then**
11: qa = qa+$|c_i|$ {all vehicles in c_i join in qa}
12: **else**
13: qa = qa+$|c_i| \cdot \Delta dur/dur_{c_i}$ and **break**
14: **end if**
15: **end if**
16: **end if**
17: **end for**
18: **return** qa

more realistic. Second, the future case with $t_o > 0$ is added to support a look-ahead decision process.

3) Intuition: After the aggregation, the non-uniformly distributed elements in traffic flow can be categorized into three types: *anticipated queue*, *platoon*, and *minor cluster*. A cluster c is considered a platoon if $|c| \geq th_n$ (a pre-specified size threshold) and $flow_c \geq th_f$ (a pre-specified flow rate threshold). Otherwise, c is a minor cluster.

For maximizing the average service rate [8], a cluster has a high priority to be serviced if it has a high flow rate over a long time duration. Long queues and platoons satisfy this requirement. Minimization of vehicle queues [3], [7] is a straightforward means for achieving green waves, since a zero queue means an ideal optimization. However, clearing all existing queues will not necessarily lead to this optimal result, since queues certainly impose delays and the presence of small queues can lead to excessive switching costs. The concept of clearing anticipated queues [8] provides a stronger basis for minimizing vehicle queues, but still requires an existing queue to be applied. More generally, a strategy that focuses strictly on clearing queues is myopic and misses opportunities to keep vehicles moving.

Based on the intuition behind "green waves", we shift the focus from clearing vehicle queues to maintaining movement of vehicle platoons. First, an ideal optimization might be achieved by responding to incoming platoons in time, based on a look-ahead decision process. Second, implicit coordination might be naturally achieved by considering platoons of vehicles as "messages" that pass between neighboring intersections. Third, large time gaps between platoons might provide natural phase switchback times for the traffic light.

From a scheduling viewpoint, the intersection can seen as a single machine, processing jobs (i.e., vehicle clusters) that arrive over time. In this sense, the decision policies described below can be seen as "look-ahead" scheduling heuristics that consider platoon clusters as critical jobs.

C. Action Selection

As indicated earlier, the critical decisions that must be made at a given decision point are (1) whether or not to extend the current phase, and (2) if so, by how much. To make these decisions, PBSS builds on the anticipated queue clearing strategy of [8]. Once a decision is made to initiate a given phase, the anticipated queue associated with this phase is always cleared to the extent permitted by G_{max}, where the queue count qa is obtained using Alg. 1 with $t_o = -t_{ge}$. Thus, the central question left unanswered is whether or not to extend the phase when there is no queue to be serviced.

We focus on the design of platoon-based selection policies to address this question, based on the intuition in Section IV-B.3. Basically, a platoon-based heuristic tries not to delay and interrupt the moving of each platoon in the horizon. If there are vehicles in front of a platoon, it would be preferable if they are discharged before the platoon arrives at the intersection but whether this is possible depends on the competing traffic situation. The look-ahead horizon is the departure time of the platoon.

Two platoon-based selection policies are defined below: platoon-based extension (PBE) and platoon-based squeezing (PBS). Each is structured to produce a t_{ext} value and they are applied in succession until a value of $t_{ext} > 0$ is returned. If all policies return 0, then the current phase is terminated. In this paper, we assume that PBE is applied first.

1) Platoon-based extension: The PBE policy checks to see if there is a platoon on the road currently being serviced and, if there is, decides whether to extend the current green phase to service that platoon. Consider the example in Figure 1, which shows the clusters queued and approaching an intersection along the road currently being serviced (LNK_g) and the road that currently has a red light (LNK_r). On LNK_g, there is no existing queue (it has already been cleared). For C_g, there is an approaching platoon cg, with size $|cg|$, arrival time (at the intersection) arr_{cg}, and depature time dep_{cg}, and a set of minor clusters between the intersection and the platoon having a total of n_{bg} vehicles. On LNK_r, there is an existing queue of size qn_r, and C_r contains a total of n_r vehicles.

Given the presence of a platoon cg on LNK_g, Algorithm 2 determines whether the green time should be extended until dep_{cg}. PBE considers two possible futures:

1) The current phase is terminated immediately, the anticipated queue qa_r on LNK_r is cleared, the platoon cg and all preceding vehicles on LNK_g are then cleared, and finally the remaining vehicles n_{rem} on LNK_r are cleared.
2) The current phase is extended to clear all vehicles on LNK_g, and then all vehicles on LNK_r are cleared. The choice that minimizes cumulative delay is taken.

In more detail, Algorithm 2 proceeds from the viewpoint of choice (1). Lines 1–2 compute the anticipated queue qa_r and its estimated clearing time t_{qc}. In order to not delay the platoon, the next green phase on LKN_g must start before t_{idle} (Line 4), which in turn implies that the light must switch to LNK_r before t_{switch} to accommodate the yellow phases

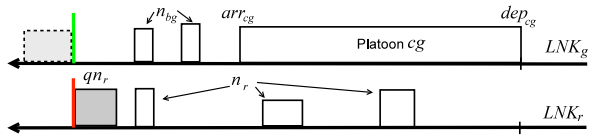

Fig. 1. Example for platoon-based extension (PBE)

Algorithm 2 Platoon-based extension (PBE) policy

1: qa_r = Alg. 1 with $t_o = Y$ and (qn_r, C_r)
2: $t_{qc} = \max(QCT(qa_r, 0), G_{min})$ {using Eq. 1}
3: $n_{bg} = \sum_{i=1}^{|C_g|-1} |c_i|$ {vehicles before the platoon cg}
4: $t_{idle} = arr_{cg} - QCT(n_{bg}, 0)$ {to next green on LNK_g}
5: $t_{switch} = t_{idle} - 2 \cdot Y$ {latest switching time to LNK_r}
6: $t_{conf} = t_{qc} - t_{switch}$ {conflicting time for the platoon}
7: **if** $t_{conf} > 0$ **then**
8: $n_r = \sum_{i=1}^{|C_r|} |c_i| + qn_r$ {total vehicles on LNK_r}
9: $n_{rem} = n_r - qa_r$ {remaining vehicles on LNK_r}
10: $\delta_r = qa_r \cdot dep_{cg} + n_{rem} \cdot (t_{switch} - t_{SL})$
11: $\delta_g = n_{bg} \cdot t_{idle}/2 + (n_{bg} + |cg|) \cdot (t_{conf} + t_{SL})$
12: **if** $\delta_g > \delta_r$ **return** $t_{ext} = dep_{cg}$
13: **end if**
14: **return** 0

(Line 5). Line 6 computes the difference (t_{conf}) between the time needed to clear qa_r and t_{switch}. If $t_{conf} > 0$, the platoon will be delayed, and the switching delay is evaluated.

In Lines 8–9, the number of vehicles n_{rem} remaining on LNK_r after queue clearing is calculated. The delay with respect to choice (2) is evaluated in Lines 10–12, where δ_r and δ_g represent respectively the gain and loss in cumulative delay on LNK_r and LNK_g for all vehicles in the horizon. In Line 10, the first term specifies the gain given that qa_r will clear dep_{cg} ticks earlier (i.e., from $(dep_{cg} + Y)$ to Y); and the second term specifies the gain from the remaining n_{rem} vehicles on LNK_r departing t_{switch} ticks earlier, i.e., from $(dep_{cg} + Y + t_{qc})$ to $(t_{conf} + dep_{cg} + Y)$, as well as the latter one requires additional t_{SL} ticks more than the former one for the start-up process. In Line 11, the first term gives the delay for n_{bg} (between $[0, t_{idle}]$) based on an averaged estimation; and the second term gives the delay for the $(n_{bg} + |cg|)$ vehicles on LNK_g that depart $(t_{conf} + t_{SL})$ ticks later.

Any platoons on LNK_r are treated as minor clusters. But the constituent vehicles still contribute to estimation of the total delay $\delta_g + \delta_r$ by belonging to either qa_r or n_{rem}.

2) Platoon-based squeezing: The PBS policy checks to see if there is a platoon on the road that currently has a red light. If so, it determines if it can extend the current phase so that the transition to green is timed to best serve that platoon. Figure 2 shows an example where, on LNK_g, the queue has just cleared the intersection. On LNK_r, there is a queue of size qn_r, a sequence of minor clusters containing n_r total vehicles, and a platoon cr with an arrival time arr_{cr}.

Upon detection of an approaching platoon on LNK_r, PBS (Algorithm 3) decides whether to extend the current green phase for some non-conflicting period (t_{nonc} in Line 3) even when there is no existing queue on LNK_g. The intention is

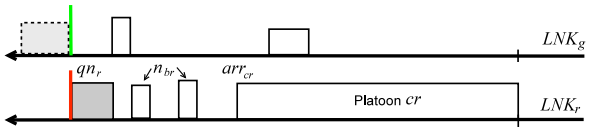

Fig. 2. Example for platoon-based squeezing (PBS)

Algorithm 3 Platoon-based squeezing (PBS) policy

1: $n_{br} = \sum_{i=1}^{|C_r|-1} |c_i|$ {arriving cars before the platoon cr}
2: $t_{qc} = \max(QCT(n_{br} + qn_r, 0), G_{min})$ {using Eq. 1}
3: $t_{nonc} = arr_{cr} - t_{qc} - Y$ {latest non-conflicting time}
4: **if** $t_{nonc} > 0$ and $t_{nonc} < (G_{min} + 2 \cdot Y)$ **then**
5: **return** $t_{ext} = t_{nonc}$
6: **end if**
7: **return** 0

to squeeze out the idle time on LNK_r without stopping the platoon. While some vehicles preceding the platoon may suffer an additional delay, shifting the available variable time may help ensure that the entire platoon, rather than only a portion, can be serviced before G_{max} is reached. In addition, downstream intersections may benefit from the creation of larger platoons with higher flow rates. If the platoon is sufficiently far away that the traffic signal can cycle through all of its phases and still switch in time to service the platoon (Line 4), then PBS will not extend the current phase.

V. Simulation Settings

We evaluate the performance of the traffic control system in simulation using an open-source microscopic road traffic simulator, Simulation of Urban Mobility[1] (SUMO). For each instance, we calculate the mean of 100 independent runs.

Our tests use two road networks: an artery with five intersections and a grid with six intersections. All roads are one-way. The lengths (L) of all road segments are identical, and $L \in \{250, 500\}$ meters. On each road, an advance detector is located $L_{det} = L - 50$ meters from the intersection, the flow speed is $v_F = 9.5$ meters/second, the saturation flow rate is $flow_{SAT} = 1/3$ vehicles/second, and the start-up loss time is $t_{SL} = 3$ seconds. For each intersection, G_{min} and G_{max} are respectively 5 and 55 seconds, and the yellow time is $Y = 5$ seconds.

The arterial network is shown in Figure 3, where the intersection **O** is considered as a bottleneck, and vehicles are mainly dispersed on the arterial road with four downstream intersections **A**–**D**. The intersection **O** provides a switched flow for the arterial road. It is controlled by a fixed timing plan with a cycle length of 70 seconds: 35 seconds for traffic on the artery and 25 seconds for traffic on the cross road.

Incoming traffic is divided among the roads with the proportions shown in Figure 3. No turns occur at any intersection except **O**. Initially, the proportion of traffic turning left onto the artery at **O** is $r_t = 0$. The total simulation period is one hour, and every twenty minutes the turning proportion increases as $r_t = r_t + \Delta r_t$ with $r_s + r_t = 5/16$.

[1] http://sumo.sourceforge.net

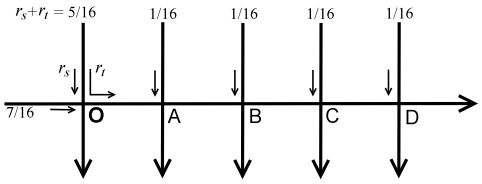

Fig. 3. The arterial network with a bottleneck intersection

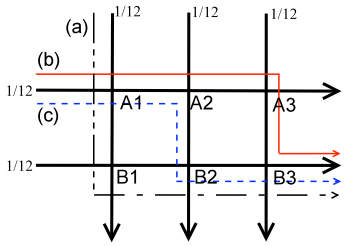

Fig. 4. 2X3 grid network

TABLE I

WAITING TIMES FOR ROADS AT INTERSECTIONS **A**–**D** ON THE ARTERIAL NETWORK: L=250, VEHICLES=1200, Δr_t=0, 1/16, 2/16

	Δr_t = 0		Δr_t = 1/16		Δr_t = 2/16	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
FIX	0.00	6.07	1.92	6.93	6.58	9.73
ASRm	27.54	22.70	27.32	23.25	28.64	24.94
AAC	6.42	7.94	6.56	8.40	7.10	9.33
PBSS	1.36	5.01	2.03	5.73	2.58	6.50

TABLE II

WAITING TIMES FOR ROADS AT INTERSECTIONS **A**–**D** ON THE ARTERIAL NETWORK: L=500, VEHICLES=1200, Δr_t=0, 1/16, 2/16

	Δr_t = 0		Δr_t = 1/16		Δr_t = 2/16	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
FIX	0.00	5.72	3.01	7.31	11.33	12.70
ASRm	19.55	17.07	20.12	18.13	21.91	19.98
AAC	4.34	6.47	4.60	6.98	4.99	7.79
PBSS	0.89	3.93	1.69	4.99	2.21	5.93

The second network, shown in Figure 4, is a grid with six intersections. Compared to the arterial network, there are many more choices of routes. For background traffic, the straight routes on the five roads each generates a minor traffic flow of $1/12$ of the total traffic. The total simulation time is one hour, and for each twenty minute period, a different major route (a, b, c) generates $7/12$ of the total traffic.

We compare PBSS to three control strategies. In the arterial network, only **A**–**D** are managed by control strategies. In the grid network, all six intersections are managed.

For PBSS, there are three parameters for aggregation. We set $th_g = 5$ seconds, $th_n = 5$ vehicles, and $th_f = 1/th_g$.

FIX is a pre-timed traffic control strategy with explicit coordination chosen for the arterial network to maximize the average speed when $\Delta r_t = 0$. All plans use a cycle length of 70 seconds. The fixed timing plan for intersections **A**–**D** when $L = 250$ uses 43/17 second splits and 28 second offsets between lights. For $L = 500$, the plan uses 41/19 second splits and 54 second offsets between lights.

ASRm is an adaptive strategy with no coordination between neighbors. It is modified from the automatic signal retiming (ASR) method in [13]. At the end of each cycle, the new cycle length and green time splits are adjusted using critical ratios between estimated and saturation flow rates, based on classical Webster's model. The only modification is an additional rate qn/t_{cycle}, where t_{cycle} is the previous cycle length, which is added to the flow rate on each road.

The anticipated all clearing (AAC) strategy is a self-organizing control strategy with (limited) implicit coordination. AAC is an implementation of the method in [8] that assumes the same limited view of incoming traffic that we assume for our methods. Furthermore, AAC is essentially a version of PBSS without both PBE and PBS.

VI. RESULTS

To examine the performance of each control strategy on the arterial network, we considered two metrics: the arterial waiting time ($T_{w,art}$), which includes only the arterial roads leading to intersections **A**–**D**, and non-bottleneck waiting time ($T_{w,nb}$), which includes all roads leading to these intersections. $T_{w,art}$ measures the ability of a strategy to create "green waves" on the artery, whereas $T_{w,nb}$ is the overall optimization objective for the road network.

The results in Tables I and II show that as the traffic flow from the bottleneck intersection becomes more uncertain (as Δr_t increases), the relative performance of PBSS compared to the other control methods improves.

When $\Delta r_t = 0$, no turns occur at the bottleneck intersection and the expected flow on the artery from the bottleneck does not change during the simulation. In this static case, FIX achieves perfect "green waves" on the artery. However, PBSS achieves the best overall performance when the side streets are also considered. This may be due in part to the fixed cycle length of the fixed timing plan. PBSS can adapt the length of its cycle to suit the traffic demand. Compared to AAC and ASRm, PBSS achieves smoother traffic corridors, as indicated by much smaller $T_{w,art}$ values.

When Δr_t increases, $T_{w,art}$ also increases, since vehicles are arriving on the artery from the bottleneck during both green phases of the intersection. Performance of FIX degrades the most as Δr_t increases. The fixed timing plan does not adapt to the change in demand, so waiting time on the artery jumps, though waiting time on cross streets remains the same. Both PBSS and AAC degrade much more gently, with PBSS performing better due to the use of platoons in the look-ahead horizon for avoiding myopic decisions. The horizon can be short (e.g., L=250), due to the leverage from rolling decisions. ASRm can absorb these changes, although without coordination, its performance is always poor.

The advantage of PBSS over AAC also provides a demonstration on the *slower-is-faster* effect [7], a counterintuitive, but practically relevant effect in many queuing systems. A study on uniform arrival flows [7] suggests one possible reason for this effect, i.e., a road with a small utilization might wait to have enough vehicles in order to balance the

TABLE III

PERFORMANCE OF PBSS AT INDIVIDUAL INTERSECTIONS ON THE GRID NETWORK AND PERCENT IMPROVEMENT OVER AAC AND ASRM: L=500, VEHICLES=1200

	PBSS		Gain over AAC		Gain over ASRm	
	$V_{n,i}$	$T_{w,i}$	$V_{n,i}$	$T_{w,i}$	$V_{n,i}$	$T_{w,i}$
A1	6.96	5.74	1.3%	0.0%	15%	76%
A2	7.93	3.15	7.7%	15.5%	30%	203%
A3	8.01	2.90	7.7%	24.0%	26%	207%
B1	7.89	3.29	5.5%	5.2%	11%	85%
B2	7.81	3.69	4.4%	0.1%	17%	107%
B3	7.88	3.51	7.0%	12.0%	26%	163%

efficiency loss caused by switching lights. Our work is based on non-uniform arrival flows, which is more realistic. AAC does not allow idle time if queues are available, whereas both PBE and PBS allow idle times in the presence of platoons.

The grid network provides more dynamic flow patterns than the arterial network. Unlike the arterial network, where intersections **A**–**D** all had similar flows, the grid network has both heavily and lightly loaded intersections. Changing the route from (a) to (b) changes the dominant flow for the intersections A1 and B3. Changing from (b) to (c) could simulate rerouting after an accident near the intersection A3.

We examined the speed ($V_{n,i}$) and waiting times ($T_{w,i}$) averaged over the input links to each intersection. We did not include a fixed timing plan in this simulation due to the high variability of traffic flows. The results, shown in Table III, give a closer view of the performance at each intersection. PBSS decreased $T_{w,i}$ more than 10% over AAC for three intersections, and much more for ASRm. The performance difference on various intersections might also be useful for studying traffic patterns. For PBSS, A1 performs much worse than B3, considering the similar traffic demands. One possible reason is that B3 benefits from receiving more platoons than A1, since an intersection might be able to play the role of forming platoons for its downstream intersections.

VII. CONCLUSIONS

In this paper, we described a self-scheduling approach to real-time, traffic signal control. In our model each traffic signal is controlled by an independent, self-interested agent that operates with a limited local view of incoming traffic. To provide an efficient basis for determining whether to extend or terminate the current signal phase, sensed traffic data is aggregated to characterize higher-level traffic flow components such as platoons and anticipated queues. Two platoon-based policies were then defined that use this higher level representation of traffic flows as look-ahead guidance, for making difficult decisions on whether a phase should be extended if there are idle times. We compared our method, PBSS, to pre-timed plans and two adaptive strategies, one based exclusively on queue-clearing, and another based on Webster's model. For two traffic networks with dynamic vehicle flows, our approach resulted in the best performance, achieving both good control at bottleneck intersections as well as coordination of vehicle flows, including the establishment of "green waves". Our research indicates that "platoons" can be used in a look-ahead horizon as patterns for facilitating implicit coordination.

There are several aspects of the proposed self-scheduling model that warrant further investigation. One issue concerns the development of more sophisticated policies that search in the aggregate representation space and explicitly evaluate tradeoffs, where some aggregate patterns might be utilized for reducing the search effort. A second aspect involves how to combine multiple selection policies. In principle, which policies to apply and their priority order might be adjusted by learning algorithms as knowledge is accumulated for each intersection. A final issue is to explore the potential benefits of incorporating explicit coordination mechanisms.

ACKNOWLEDGEMENTS. This work was supported in part by a grant from the Hillman Foundation and the Robotics Institute, Carnegie Mellon University.

REFERENCES

[1] D. Schrank, T. Lomax, and S. Turner, "Annual urban mobility report," tech. rep., Texas Transportation Institute, Texas A&M University System, TX, 2010.

[2] T. H. Heung, T. K. Ho, and Y. F. Fung, "Coordinated road-junction traffic control by dynamic programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 341–350, 2005.

[3] D. I. Robertson and R. D. Bretherton, "Optimizing networks of traffic signals in real time - the SCOOT method," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 11–15, 1991.

[4] P. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," *Transportation Research Part C-Emerging Technologies*, vol. 9, no. 6, pp. 415–432, 2001.

[5] I. Porche and S. Lafortune, "Adaptive look-ahead optimization of traffic signals," *ITS Journal*, vol. 4, no. 3-4, pp. 209–254, 1999.

[6] C. Gershenson, "Self-organizing traffic lights," *Complex Systems*, vol. 16, no. 1, pp. 29–53, 2005.

[7] D. Helbing and A. Mazloumian, "Operation regimes and slower-is-faster effect in the control of traffic intersections," *European Physical Journal B*, vol. 70, no. 2, pp. 257–274, 2009.

[8] S. Lämmer and D. Helbing, "Self-control of traffic lights and vehicle flows in urban road networks," *Journal of Statistical Mechanics: Theory and Experiment*, p. P04019, 2008.

[9] G. Slager and M. Milano, "Urban traffic control system using self-organization," in *IEEE International Conference on Intelligent Transportation Systems*, (Madeira Island, Portugal), pp. 255–260, 2010.

[10] S. El-Tantawy and B. Abdulhai, "An agent-based learning towards decentralized and coordinated traffic signal control," in *IEEE International Conference on Intelligent Transportation Systems*, (Madeira Island, Portugal), pp. 665–670, 2010.

[11] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[12] S. G. Shelby, "Single-intersection evaluation of real-time adaptive traffic signal control algorithms," *Transportation Research Record*, vol. 1867, pp. 183–192, 2004.

[13] S. Cheng, M. Epelman, and R. Smith, "CoSIGN: A parallel algorithm for coordinated traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 551–564, 2006.

[14] C. Papadimitriou and J. Tsitsiklis, "The complexity of optimal queuing network control," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 293–305, 1999.

[15] S. F. Smith, "Is scheduling a solved problem?," in *MultiDisciplinary Scheduling: Theory and Applications*, pp. 3–17, Springer, 2005.

[16] A. Sharma, D. Bullock, and J. Bonneson, "Input-output and hybrid techniques for real-time prediction of delay and maximum queue length at signalized intersections," *Transportation Research Record*, vol. 2035, pp. 69–80, 2007.

[17] T. Nagatani, "Vehicular traffic through a sequence of green-wave lights," *Physica A*, vol. 380, pp. 503–511, 2007.