Self-Scheduling Agents for Real-Time Traffic Signal Control

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

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
{xfxie,gjb,sfs,zbr}@cs.cmu.edu

Technical Report CMU-RI-TR-11-06

February 4, 2011

Abstract

Optimizing traffic signal control in real-time to respond to changing vehicle flows is a challenging problem. As the size of the road network increases, the exponential growth in joint signal timing and traffic states presents a difficult computational barrier to determining effective traffic light signalization. Furthermore, the dynamic nature of traffic flows makes reliable prediction possible only over a limited time horizon, and hence necessitates continual re-computation and adaptation of computed solutions. 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-uniform distributed nature of road traffic flows. Such aggregation of vehicles into *anticipated queues* and *platoons* enables the real-time computation of signal timing policies that incorporate greater look-ahead than traditional queue clearing strategies and better promote the establishment of "green waves" where vehicles flow through the road network without stopping. We present simulation results on some dynamic traffic scenarios that demonstrate the leverage that our approach provides over two state-of-the-art self-organizing approaches.

1 Introduction

Self-scheduling systems have gained increasing attention in recent years [21], 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. Such systems do not attempt to produce optimal solutions and no attempt is made to globally coordinate the decisions of individual agents. Instead the emphasis is on adapting effectively to dynamically changing execution circumstances. As execution proceeds, each agent makes independent decisions based on currently available local information that promote effective emergent global behavior.

In this paper, we consider the design of a self-scheduling system for real-time traffic signal control. Traffic signal control is an important practical problem. A recent study on urban mobility in the USA [23] indicates that a given traveler now spends an average of $750 annually in wasted fuel and productivity due to traffic congestion. Moreover, since this wasted fuel and time corresponds directly to the percentage of time drivers spend idling in traffic, there are additional 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, there continues to be a recognized gap in technologies that can respond efficiently and effectively to real-time traffic demand changes [17]. The problem is quite challenging. On one hand, the number of joint timing plans and traffic conditions is huge for even an individual intersection [13, 19] and grows exponentially with the size of the network [5]. On the other, the non-linear dynamics of a switched traffic network [12] 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 approach. 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 immediate incoming road segments). Central to our approach is an aggregate representation of traffic flows as clusters of *anticipated queues* and *platoons*. These aggregate patterns provide a basis for real-time signal control policies that incorporate greater look-ahead than the queue clearing strategies utilized by prior work in self-organizing traffic signal control systems (e.g., [10]). We present simulation results on two traffic networks with dynamic vehicle flows that demonstrate the performance benefit of our extended look-ahead approach.

The remainder of this paper is organized as follows. We first briefly review prior research in adaptive traffic signal control. Next, we formulate the real-time traffic signal control problem to be addressed in this paper. Our basic approach to modeling traffic flows and making traffic signal control decisions within each agent is described in Sections 3 and 4 respectively. In Sections 5 and 6 we

present experimental results that indicate the performance characteristics of the proposed approach. Finally, in Section 7 we summarize and indicate directions for future research.

2 Related Work

Multi-agent systems provide a natural framework for addressing the real-time, traffic signal control problem, given the distributed nature of traffic flow information [1, 2, 8, 10, 6, 7, 24]. The overall complexity of the problem can be decomposed into two components: network-level coordination and agent-level optimization. Different approaches have placed different emphases on these two solution components.

2.1 Traffic Light Coordination

One sign of good coordination between traffic lights is the presence of a "green wave" of vehicles, progressing through sequences of lights without unnecessary stopping. However, in such a circumstance there is also a cost of waiting time for conflicting traffic flows. Thus, overall optimization is an essential requirement.

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. In most conventional traffic control systems, cycle timing plans are hand built and fixed. One set of adaptive approaches uses explicit coordination strategies to dynamically adjust the *cycle lengths*, *phase split times*, and/or *offsets* along a sequence of lights on a given route over time. In some cases [16], these adjustments are decided offline and periodically installed, and in others (e.g., [14]), they are computed online as circumstances warrant. In both cases, however, these methods typically require a degree of stability in traffic flows over time to enable coordinating agents to build knowledge of current traffic flow patterns.

Optimization-based coordination methods, alternatively, try to achieve a system-wide agreement on the choices from the state space formed by tentative signal timing segments, phases, and/or plans that are proposed by individual agents. The coordination problem might be modelled as a distributed constraint satisfaction problem [6], or as game playing [2, 5]. From a real-time perspective, however, these methods frequently require a long time to converge to an equilibrium.

Coordination between fully autonomous agents can also be achieved implicitly through joint perception of the shared environment in which they operate. Several reinforcement learning approaches have been developed based on this assumption [24, 15]. These approaches are also often slow to converge and difficult to apply if traffic flows are changing frequently. Another implicit approach to network coordination is through use of self-organizing methods, which try to use simple clues of global patterns in the shared environment [1, 8, 10, 11].

2.2 Agent-Level Optimization

At the individual agent level, the control problem is a single intersection, and the objective is to make the best local decisions possible, given the information that is locally available.

General optimization methods [13], especially dynamic programming [19, 3], have been used for controlling a single intersection. Normally, an exponential state space is formed by dividing the signal planning horizon into discrete time steps. The evaluation and transition of states are based on dynamic traffic models, but the inherent traffic patterns are ignored. To make the search tractable in real time, these approaches often resort to aggregate time steps (e.g., 5 seconds [19]) or use approximate techniques [3], both of which can greatly compromise performance [3]. A further difficulty is the limited lifetime of such optimal solutions, if the underlying models are only reliable in the short term.

Self-organizing agent approaches to the signal control problem have also received considerable attention [1, 8, 10]. In [8], a self-organizing traffic light is controlled by thresholding 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 unneccessary stopping. In [1], vehicles contribute history-based "credits" for pass through traffic lights. 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 [10]. The importance of serving queues can be traced to the study of queuing systems [4], although their interests are mainly in the stability of scheduling policies. The concept of an "anticipated queue" further extends the concept of a queue to one that might include arriving vehicles, including "platoons". The approach we propose incorporates and builds on this idea.

2.3 Self-Scheduling Agents

We take a self-scheduling agent approach [21] to traffic signal control in this paper. Under the law of bounded rationality [20, 9], a self-scheduling agent is realized in two essential layers. First, the structural information of the local environment is formed into an aggregate model, which captures some critical flow components, including *anticipated queues* and *platoons*, in non-uniform distributed traffic flows. Second, each agent makes fast-and-frugal scheduling decisions based on those critical flow components while respecting current traffic state, in order to adapt itself quickly in real time.

Compared to the above described agent systems based on general optimization, the self-scheduling agent does not attempt to make optimal schedules by searching the huge state space formed by primary (i.e., detection level) traffic flow information. Compared to purely self-organizing agents, the self-scheduling agent attempts to exploit a limited look-ahead search based on the aggregate model.

In principle, there is no explicit global information required by a self-scheduling agent. However, there is also no reason not to utilize any communicated or glob-

ally available information if it is useful. In this paper, we restrict attention to a model that relies strictly on locally sensed traffic flow information.

3 Problem Formulation

The traffic signal control problem focuses on maximizing the flow of traffic through a road network gated by traffic lights. The challenge is in assigning the green phases of each light such that the maximum number of vehicles traverse the network in a given time while observing safety and fairness constraints. While there are a variety of ways to measure the flow of the network, we will use the average speed and the average wait time of the vehicles in a given road network over a specific time period. The faster the speed and the shorter the wait time, the better the flow is in the network.

Each traffic light controls an intersection, which consists of a set of roads from which vehicles enter the intersection and a set of roads from which vehicles depart the intersection. Traffic lights function by cycling through their light phases, with green permitting vehicles to travel through the intersection, yellow requiring them to slow down and prepare to stop prior to the intersection, and red forcing vehicles to wait at the intersection. Each period that a light has a particular illumination, e.g., green for north-south traffic and red for west-east traffic, is defined as a phase.

There are safety constraints—if a light is green or yellow for one incoming road, then the light must be red for all other conflicting roads—and fairness constraints—all incoming roads are guaranteed a green light for a minimum time window. The yellow light phase runs for a fixed duration τ_y, while the green and red lights have variable duration between a set minimum (τ_g^{min}) and maximum (τ_g^{max}). This variable time is what the traffic light controller adjusts to maximize flow. For example, if there is a heavier traffic on one incoming road than on others, then the controller could increase overall flow by using the variable time to extend the green light for the road with a denser flow of vehicles, though this would result in a longer red time for other incoming roads. As the flows change in the network or as volume increases, frequent recomputation to allocate the variable time may be necessary to service the incoming roads.

For this paper, we will assume that all intersections consist of two one-way roads, one with traffic going from west to east and the other with traffic going from north to south. Each road starts at either an intersection or an entry point into the network, and each road ends at either an intersection or an exit point from the network. Each road has a fixed length (L). While this simplifying assumption aids in the description of the proposed algorithm, it is not a limitation. The algorithm naturally extends to two-way road networks.

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, where $L_{det} < L$. In the next section, we will describe our proposed algorithm, platoon-based self-scheduling (PBSS), showing how lookahead information provided by each pair of sensors on each incoming road can be leveraged

to constantly recompute the allocation strategy for the variable time, adapting in real time as changes in the traffic flow and volume arise. We contend that, by having all lights self-schedule using PBSS effective overall flow for the entire road network can be achieved.

4 Platoon-Based Self-Scheduling

The basic idea behind PBSS is to repeatedly collect data over a window of time between decision points, aggregate the data based on their proximity into groups called *clusters*, and then use the clusters to determine whether to extend the current phase, i.e., allocate the variable time, or transition to the next phase. In addition, the length of time until the next decision point is computed and used for the next iteration of the algorithm. This loop is repeated indefinitely.

4.1 Detection

Between decision times, we collect data about vehicles over a detection window. Vehicles are tracked by periodically sampling the upstream sensor to see how many vehicles have passed over it since the last sampling. While the sampling time may be irregular or larger, in this work we assume a sampling period of one second. Sampling over one period results in a tuple containing the start time of the sample period (τ_{ps}), the duration of the period (τ_{pd}), the end time of the period ($\tau_{pe} = \tau_{ps} + \tau_{pd}$), the number of vehicles that were counted (n_{pc}), and the flow rate of vehicles ($q_{pc} = n_{pc}/\tau_{pd}$). Rather than using the absolute start and end times, we use the offset of the tuple from the intersection, so $\tau_{ps}(t) = \tau_{ps} - t - L_{det}/v_f$, where v_f is the expected vehicle speed, or free flow speed. This offset $\tau_{ps}(t)$ is the expected amount of time it will take for the first car counted in this sampling period to reach the intersection. A positive offset τ_{ps} at time t implies that the vehicles represented in the tuple have not yet reached the intersection. Tuples from each detection period where $n_{pc} > 0$ are added to an ordered sequence to be processed at the next decision time.

In addition to new vehicles being tracked, vehicles crossing through the intersection are monitored via the stop-line detector. We maintain a count of the vehicles in the queue at the intersection for each incoming road (n_{qn}). Each time a vehicle crosses the stop-line detector, it is removed from the appropriate queue count. During the decision point at the end of the detection window, new queue counts are generated by incorporating new vehicles expected to join the existing queue. Queue counts are carried over to the next detection window.

4.2 Aggregation

Once we reach the next decision point, the tuples collected over the detection window are aggregated into clusters and added to an ordered sequence (S) which may contain previously generated clusters from previous decision points. As we will show in section 4.4, clusters help both in identifying significant blocks of demand and in reducing the complexity when deciding what action to take. The

clusters are formed by mapping over the tuples and merging them when they are within a specified threshold duration (c_{thc}) of each other. The resulting tuple uses the minimum start time of the merged tuples for the new start time (τ_{ps}), the sum of the durations for the new duration (τ_{pd}), and the sum of the counts for the new count (n_{pc}). The end time (τ_{pe}) and flow rate (q_{pc}) are recalculated. The offset of a cluster from the intersection is calculated as before. The merged clusters are categorized into three types: *queue*, *platoon*, and *minor*. The queue clusters are those clusters that are expected to have arrived at the intersection ($\tau_{ps} \leq 0$). Platoon cluster are not expected to have arrived at the intersection ($q_{pc} > 0$), have a count of cars larger than a threshold ($n_{pc} > c_{thpc}$), and have a flow rate greater than a threshold ($q_{pc} > c_{thpd}$). The rest of the clusters are classified as minor clusters.

4.3 Queue Estimation

The generated clusters in S are used to anticipate the number of vehicles that are presently or in the future will be waiting in the queue at the intersection for each of the incoming roads. This anticipated queue size will be used in selecting actions described in section 4.4. A key computation in this calculation is how long it takes for a queue of vehicles at the intersection to be cleared, i.e., how long would the light need to be green for all the vehicles to go through the intersection. To determine the queue clearing time (τ_{qc}), we use Equation 1, a straightforward model presented in [18], where n_q is the number of vehicles in the queue, τ_{ge} is the elapsed time that the light has been green for this road, τ_{sl} is the amount of time lost in vehicles starting up, and τ_{sh} is the saturation headway, i.e., the time between each vehicle in a group of vehicles moving the same speed.

$$\tau_{qc}(n_q, \tau_{ge}) = \begin{cases} \tau_{sl} - \tau_{ge} + \tau_{sh} \cdot n_q & \text{if } \tau_{ge} < \tau_{sl} \\ \tau_{sh} \cdot n_q & \text{if } \tau_{ge} \geq \tau_{sl} \end{cases} \quad (1)$$

The anticipated number of vehicles in the queue is calculated using Algorithm 1. The algorithm is invoked with the current count of vehicles stopped at the intersection (n_{qn}), the elapsed green time (τ_{ge}), and the future time for which the queue should be calculated (τ_{adv}). This algorithm returns n_{qa}, the number of vehicles that are at the intersection at the specified time or are expected to arrive before the queue clears.

4.4 Action Selection

The next step in PBSS after creating the clusters is deciding what action to take. The two possible actions are to extend the current phase of the light, i.e., consume some of the variable time, or transition to the next phase. This decision is computed by trying in order a prioritized series of selection policies. If a policy returns a time, it becomes the selected action and is used for extending the current phase. If a policy returns zero, then the next policy in the series is tried. If all policies return zero, then the current phase is terminated and the

Algorithm 1 Estimate n_{qa} in the cluster sequence S

Require: Either $\tau_{ge} = 0$ or $\tau_{adv} = 0$
  $n_{qa} = n_{qn}$
  for $i = 0$ to $Size(S)$ **do**
    Obtain $\tau_{qc}(n_{qa}, \tau_{ge})$ by using Eq. 1
    if $(\tau_{ps,i} - \tau_{adv}) \leq \tau_{qc}$ **then**
      $\Delta_d = 1/\tau_{sh} - q_{pc,i}$
      if $\Delta_d \leq 0$ or $(\tau_{pe,i} - \tau_{adv}) \leq \tau_{qc}$ **then**
        n_{qa} += $n_{pc,i}$
      else
        $\Delta_t = (\tau_{qc} - (\tau_{ps,i} - \tau_{adv})) \cdot q_{pc,i}/\Delta_d$
        if $\Delta_t < \tau_{pd,i}$ **then**
          n_{qa} += $n_{pc,i} \cdot \Delta_t/\tau_{pd,i}$
          return
        else
          n_{qa} += $n_{pc,i}$
        end if
      end if
    end if
  end for

next two phases are installed, i.e., the yellow phase for the light that is green followed by the minimum duration for the current red light to be green. The next decision time is based on either the end of the extended time returned by a selection policy or the end of the second installed phase. The actual decision time is slightly less than either of those times to allow for computation time to analyze the collected data and select an action. Currently, there are three selection policies. They are, in order of priority, anticipated all clearing (AAC), platoon-based extension (PBE), and platoon-based squeezing (PBS).

4.4.1 Anticipated all clearing

The AAC policy, which is based on the policy presented in [10], checks to see if there is an anticipated queue at the current time at the intersection for the road that is being serviced, i.e., has the green light. If so, it will try to extend the green light to service the vehicles in that queue. Algorithm 2 describes how this policy is computed.

Algorithm 2 Anticipated all clearing (AAC) policy

Require: The phase index k, the elapsed green time τ_{ge}
  Estimate n_{qa} with Algorithm 1, given τ_{ge} and $\tau_{adv} = 0$
  Obtain $\tau_{ext} = \tau_{qc}(n_{qa}, \tau_{ge})$ by using Eq. 1
  $\tau_{ext} = min(\tau_{ext}, the\ remaining\ variable\ time\ at\ k)$
  return τ_{ext}

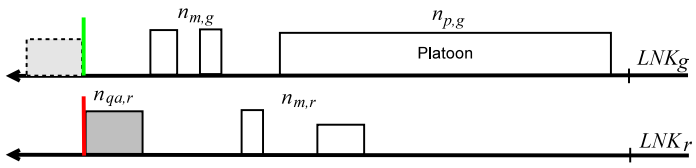

Figure 1: Example for platoon-based extension

Algorithm 3 Platoon-based extension (PBE) policy

1: Estimate $n'_{qa,r}$ with Algorithm 1, given $\tau_{adv} = \tau_y$
2: $\tau'_{qa,r} = \tau_{qc}(n'_{qa,r}, 0)$ {using Eq. 1 for τ_{qc}}
3: $\tau'_{qa,r} = max(\tau'_{qa,r}, the\ remaining\ variable\ time)$
4: $\tau_{idle,g} = \tau_{ps,g} - \tau_{qc}(n_{m,g}, 0)$ {using Eq. 1 for τ_{qc}}
5: $\Delta_\tau = (\tau'_{qa,r} + 2 \cdot \tau_y) - \tau_{idle,g}$
6: **if** $\Delta_\tau > 0$ **then**
7: $\quad n'_{m,r} = n_{m,r} + n_{qa,r} - n'_{qa,r}$
8: $\quad \delta_r = n'_{m,r} \cdot \Delta_\tau - n'_{qa,r} \cdot (\tau_{pe,g} + n'_{m,r} \cdot \tau_{sl})$
9: $\quad \delta_g = (\Delta_\tau + \tau_{sl}) \cdot (n_{p,g} + n_{m,g}) + n_{m,g} \cdot \tau_{idle,g}/2$
10: $\quad$ **if** $\delta_g + \delta_r > 0$ **then return** $\tau_{ext} = \tau_{pe,g}$
11: **end if**
12: **return** $\tau_{ext} = 0$

This policy maintains a green light in order to service vehicles that are expected to become part of the queue before the queue is able to clear. As long as the maximum green time is not reached, a vehicle joining the queue while the light is green will not have to stop at the light when it turns red.

4.4.2 Platoon-based extension

The PBE policy checks to see if there is a platoon on the road currently being serviced. If there is, then it tries to extend the current phase to service that platoon. Figure 1 shows an example where Links LNK_g has a green light and LNK_r has a red light. On LNK_g, the queue has just been serviced, and the platoon has the count $n_{p,g}$, the start time $\tau_{ps,g}$, and the end time $\tau_{pe,g}$, and the minor clusters before the platoon have the total count $n_{m,g}$. On LNK_r, there is a queue with the count $n_{qa,r}$, and minor clusters with the total count $n_{m,r}$.

As shown in Algorithm 3, the role of PBE is to decide if the green time should be extended to the end of the platoon, based on the end time of the platoon (Line 10). From Lines 1 to 3, the anticipated queue on LNK_r is estimated, given an advance time τ_y (yellow time). Then, the maximum idle time $\tau_{idle,g}$ that the platoon will not be stopped is estimated in Line 4. Line 5 gives the non-conflicting time Δ_τ, based on the total switching time. The decision is based on an estimation of the total delay caused by the switching operation (Lines 7 to 10), in which δ_r and δ_g are the delay difference on LNK_r and LNK_g for the clusters in the knowledge horizon, respectively.

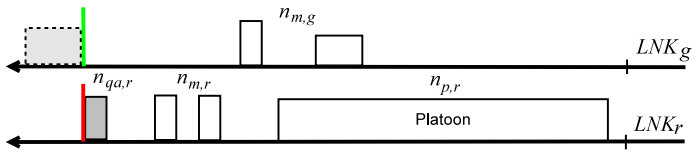

Figure 2: Example for platoon-based squeezing

Algorithm 4 Platoon-based squeezing (PBS) policy

1: $\tau'_{qa,r} = \tau_{qc}(n_{qa,r} + n_{m,r}, 0)$ {using Eq. 1 for τ_{qc}}
2: $\tau'_{qa,r} = max(\tau'_{qa,r}, the\ remaining\ variable\ time)$
3: $\tau_{idle,r} = \tau_{ps,r} - \tau'_{qa,r} - \tau_y$
4: **if** $\tau_{idle,r} < (\tau_{sg}^{min} + 2 \cdot \tau_y)$ **then**
5: **return** $\tau_{ext} = \tau_{idle,r}$
6: **end if**
7: **return** $\tau_{ext} = 0$

Unlike AAC, PBE might choose to be idle, i.e., maintain a green light for an incoming road even if there is not a queue of vehicles, in order to serve a forthcoming platoon. It may be the case that having the wait introduced by having the platoon stopped while the phases cycle around may be greater than the wait introduced by idling.

4.4.3 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 for that road is timed to best serve that platoon. Figure 2 shows an example where, on LNK_g, the queue just left the intersection, and there are minor clusters with the total count $n_{m,g}$. On LNK_r, there are a queue with the count $n_{qa,r}$, a platoon with the count $n_{p,r}$, the start time $\tau_{ps,r}$, and the end time $\tau_{pe,r}$, and minor clusters with the total count $n_{m,r}$.

As in PBE, the role of PBS is to decide whether the current green phase should be extended for the idle time in the next phase ($\tau_{idle,r}$ in Line 3) even when there is no queue available. The intention is to squeeze out the idle time without stopping the platoon. While some minor clusters ($n_{m,g}$) may suffer an additional delay, shifting the available variable time may help ensure that the entire platoon ($n_{p,r}$), rather than only part of the platoon, can be serviced before the maximum green time is reached. In addition, downstream intersections may benefit from platoons with higher flow rates.

If the platoon is sufficiently far away that the traffic signal can transition through all of the phases in the cycle and still switch in time to service the platoon, then the policy will not extend the current phase. In the current series of selection policies, where PBS is the lowest priority policy, if the phase is not extended, then we transition to the next phase.

Unlike PBE, PBS does not make its decision based on a quality measurement, nor does it use any information about the number of vehicles in the platoon or the end time of the platoon.

5 Experiments

We evaluate the performance of the platoon-based self-scheduling traffic control system in simulation using an open-source microscopic road traffic simulator, Simulation of Urban Mobility[1] (SUMO).

Our tests use two road networks: an artery with five intersections and a grid with six intersections. All roads are one-way. To measure performance, we consider two evaluation metrics: average speed (total travel time for a vehicle divided by its total travel distance) and the average waiting time per vehicle. For each instance, we calculate the mean of ten independent runs.

For each network, the distance between nodes (L) is identical, and $L \in \{250, 500\}$ meters. On each road, an advance detector is located $L_{det} = L - 50$ meters from the intersection. The minimum green time is $\tau_g^{min} = 5$ seconds, the maximum green time is $\tau_g^{max} = 55$ seconds, and the yellow time is $\tau_y = 5$ seconds.

The first network is an arterial road with five cross streets. Incoming traffic is divided among the roads with the proportions shown in Figure 3. No turns occur at any intersection except **O**, where traffic from the cross street turns onto the artery with probability $r_t/(r_s + r_t)$. Initially, $r_t = 0$. The total simulation period is one hour, and every twenty minutes the turning proportion increases as $r_t = r_t + \Delta r_t$ with $r_s + r_t = 5/16$.

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. We use only 8 routes: the straight routes on each road, which had minor traffic flows generating 1/12 of total traffic each, as well as three routes with major traffic flows. 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 evaluate three versions of our self-scheduling control method. The main version, PBSS, uses both Platoon-based Extension (PBE) and Platoon-based Squeezing (PBS). PBSSe uses only PBE and PBSSs uses only PBS. The three flow model parameters, flow velocity (v_f), start-up loss time (τ_{sl}), and saturation headway (τ_{sh}), can be estimated from historical traffic data [18]. We use values of $\tau_{sl} = 3$ seconds, $\tau_{sh} = 3$ seconds, and $v_f = 0.95 v_{max}$, in which v_{max} is the speed limit of 10 meters per second. For the aggregation model parameters, we set $c_{thc} = 5$ seconds, $c_{thpc} = 5$ vehicles, and $c_{thpd} = 1/c_{thc}$.

We compare PBSS to a fixed timing plan and two existing adaptive strategies. The Anticipated All Clearing (AAC) policy can be seen as an implementation of the method in [10] that assumes the same limited view of incoming traffic that we assume for our methods. AAC can also be seen as a version of PBSS without either PBE or PBS and $c_{thc} = 0$.

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

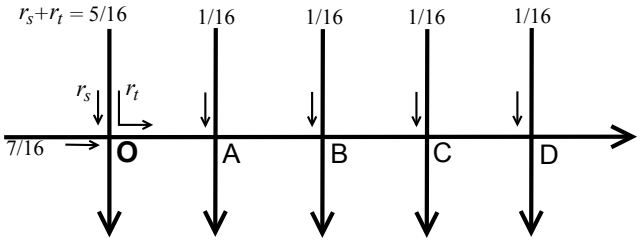

Figure 3: 1X5 arterial network

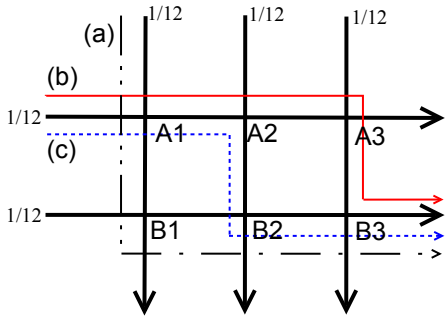

Figure 4: 2X3 grid network

SOTL is the "Sotl-phase" version of the self-organizing traffic light controller presented in [8]. Each traffic light controller keeps a count (κ_i) of the number of vehicles times time steps approaching a red light within a given detection region. The controller turns the light green if κ_i reaches a threshold θ, and then sets $\kappa_i = 0$. The controller constrains the minimum green time by τ_g^{min} to prevent switching the light too quickly. We use the default parameter values $\theta = 41$ and $\tau_g^{min} = 20$ from [8]. We also set the detection region to ten seconds, which is similar to the setting in [8].

We chose a fixed signal timing plan (FIX) for each road network to maximize the average speed when $r_t = 0$. All plans use a cycle length of 70 seconds and control intersection **O** using a timing plan with splits of 35 seconds to traffic on the artery and 25 seconds to cross traffic, consistent with the proportion of traffic arriving at the intersection. 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.

We conduct tests on three scenarios. The first scenario uses the arterial network and allows all intersections to be adaptively controlled. The second

Table 1: Overall performance on the arterial network: L=250, Vehicles=1200, Δr_t=0, 1/16, 2/16

	$\Delta r_t = 0$		$\Delta r_t = 1/16$		$\Delta r_t = 2/16$	
	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$
FIX	6.70	17.883	6.63	19.15	6.11	24.19
PBSS*	6.39	18.47	6.47	18.39	6.43	19.78
SOTL	4.49	37.00	4.45	40.16	4.14	48.40
AAC	6.24	14.51	6.22	15.55	6.14	16.88
PBSS	6.86	12.52	6.86	13.15	6.85	13.52

Table 2: Overall performance on the arterial network: L=500, Vehicles=1200, Δr_t=0, 1/16, 2/16

	$\Delta r_t = 0$		$\Delta r_t = 1/16$		$\Delta r_t = 2/16$	
	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$
FIX	7.77	17.01	7.72	18.51	7.09	25.35
PBSS*	7.65	17.34	7.64	18.78	7.69	18.41
SOTL	6.03	34.95	5.81	39.78	5.54	47.11
AAC	7.55	13.36	7.56	14.18	7.50	15.46
PBSS	7.97	12.40	7.93	13.41	7.89	14.45

scenario uses the arterial network, but intersection **O** is controlled by a fixed timing plan for all controllers. The fixed plan gives 35 seconds to arterial traffic and 25 seconds to cross traffic. This tests the ability of each controller to disperse traffic from an uncontrolled bottleneck. The third scenario uses the grid network and allows all intersections to be adaptively controlled.

6 Results

We began by evaluating the overall performance when each method was allowed to control all five intersections on the arterial network shown in Figure 3 with $L \in \{250, 500\}$. In addition to the methods described above, we also evaluated PBSS*, a version of PBSS where intersection **O** is controlled by a fixed timing plan. The results, shown in Tables 1 and 2, show that PBSS produced the highest average speed ($V_{n,all}$) and the lowest average waiting time ($T_{w,all}$) regardless of changing demands from intersection **O**.

The difference between PBSS* and PBSS indicates that the self-scheduling agents can control a bottleneck intersection better than a fixed signal timing plan, possibly because it can adapt the cycle length in real time. As Δr_t increases, the average speed of PBSS decreases more gradually than for the other methods. Though FIX outperforms PBSS* with $\Delta r_t = 0$, PBSS* has better relative performance when $\Delta r_t = 2/16$.

PBSS performs well when the detection length is short ($L_{det} = 200$ meters for $L = 250$ meters). A vehicle following the speed limit can pass through in about

Table 3: Waiting times for roads at intersections **A**-**D** on the arterial network: L=250, Vehicles=1200, Δr_t=0, 1/16, 2/16

	$\Delta r_t = 0$		$\Delta r_t = 1/16$		$\Delta r_t = 2/16$	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
FIX	**0.00**	6.01	**1.95**	7.02	6.68	9.78
SOTL*	29.32	22.74	32.28	25.38	41.03	32.00
AAC*	7.54	8.45	7.48	8.72	7.56	9.55
PBSS*	1.54	**5.11**	2.06	**5.75**	**2.50**	**6.42**

Table 4: Waiting times for roads at intersections **A**-**D** on the arterial network: L=500, Vehicles=1200, Δr_t=0, 1/16, 2/16

	$\Delta r_t = 0$		$\Delta r_t = 1/16$		$\Delta r_t = 2/16$	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
FIX	**0.00**	5.71	2.53	7.02	11.46	12.85
SOTL*	27.54	21.83	32.66	25.7	39.98	31.38
AAC*	4.41	6.54	4.82	7.14	5.00	7.82
PBSS*	0.83	**3.90**	**1.57**	**4.77**	**2.19**	**5.73**

20 seconds, which is much shorter than the maximum green time constraint for adaptive traffic lights. This means that even if a given road was very short, we might only need to extend the detection length slightly by using detectors from upstream intersections.

While these results show performance both on the artery and for controlling a bottleneck intersection (**O**), it is interesting to consider a situation where the bottleneck cannot be adaptively controlled. In this case, intersection **O** used a fixed timing plan, while the other four intersections (**A**-**D**) could be controlled by adaptive agents. To examine the effects on the roads controlled by these intersections, 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. The results, shown in Tables 3 and 4 show that as the traffic flow from the bottleneck intersection becomes more uncertain (as Δr_t increases), the relative performance of PBSS to the other methods improved.

When $\Delta r_t = 0$, no turns occurred at the bottleneck intersection, so the expected flow on the artery from the bottleneck did not change during the simulation. In this static case, the fixed timing plans forced very few stops on the artery. However, the waiting time on the cross streets was higher than for PBSS. This may be due in part to the fixed cycle length of the fixed timing plans. PBSS can optimize the length of its cycle to suit the traffic demand.

When Δr_t increases, $T_{w,art}$ also increases, since cars are arriving on the artery from the bottleneck during both green phases of the intersection. Performance of FIX and SOTL degraded the most as Δr_t increased. The fixed timing plan cannot adapt to the change in demand, so waiting time on the

Table 5: Overall performance on the grid network: L=500, Vehicles=900, 1200, 1500

	900 Vehicles		1200 Vehicles		1500 Vehicles	
	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$
SOTL	6.99	28.20	5.52	45.53	4.63	59.52
AAC	7.80	9.10	7.53	13.16	6.21	27.82
PBSSe	8.02	8.56	7.74	12.23	6.26	27.09
PBSSs	8.12	9.04	7.80	13.11	6.39	26.81
PBSS	8.11	8.87	7.85	12.16	6.43	25.77

artery jumped, though waiting time on cross streets remained the same. Waiting time both on and off the artery increased for SOTL, suggesting that it did not achieve good coordination. Both PBSS and AAC degraded much more gently, with PBSS performing better due to the use of platoons.

Overall, PBSS showed the best performance of the algorithms evaluated here. It was able to effectively control a bottleneck intersection as well as to disperse flows coming out of a bottleneck. As the variance in traffic patterns increased, the performance benefits became even clearer.

These results suggest that achieving "green waves" might be a pitfall for designing adaptive strategies. While "green waves" are a good sign of coordination, they do not necessarily lead to the best overall performance for a given network.

We also evaluated controllers on the grid network shown in Figure 4 with $L = 500$ and traffic demands of 900, 1200, and 1500 vehicles per hour. We did not include a fixed timing plan in this experiment due to the high variability of traffic flows. We did include two variants of PBSS to examine the effects of the platoon-based components: PBSSe uses only PBE and PBSSs uses only PBS. The results, shown in Table 5, show that the three PBSS methods outperformed AAC and SOTL. This shows that the use of platoon-based scheduling can improve performance. The results for the three PBSS methods were very similar. PBSSs has better average speed than PBSSe. PBSS, which included both components, performed best.

The grid network provided more dynamic flow patterns than the arterial network. Changing the route from (a) to (b) changed the dominant flow for the intersections A1 and B2. Changing from (b) to (c) could simulate rerouting after an accident near the intersection A3. There are also heavily loaded intersections, including A1 and B3, and light-loaded intersections, including A3 and B1.

We examined the speed ($V_{n,i}$) and waiting times ($T_{w,i}$) averaged over the input links to each intersection. The results, shown in Table 6, give a closer view of the performance at each intersection. We show the speed and waiting time for PBSS as well as the percent improvement over AAC and SOTL. PBSS decreased $T_{w,i}$ more than 10% over AAC for three intersections, and much more for SOTL. The performance difference on various intersections might also be useful for studying traffic patterns. For PBSS, A1 performed much worse than

Table 6: Performance of PBSS at individual intersections on the grid network and percent improvement over AAC and SOTL: L=500, Vehicles=1200

	PBSS		Gain over AAC		Gain over SOTL	
	$V_{n,i}$	$T_{w,i}$	$V_{n,i}$	$T_{w,i}$	$V_{n,i}$	$T_{w,i}$
A1	6.97	5.71	1.6%	2.5%	133%	325%
A2	7.95	3.23	7.5%	12.7%	54%	314%
A3	8.00	2.91	8.0%	21.7%	33%	278%
B1	7.88	3.36	6.4%	6.4%	35%	243%
B2	7.82	3.67	4.1%	0.1%	33%	194%
B3	7.86	3.53	7.2%	11.3%	52%	294%

B3, considering they have similar traffic demands, although the arrival time durations are different. One possible reason is that B3 benefited from receiving more platoons than A1, since an intersection might be able to play the role of forming platoons for its downstream intersections.

7 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, which operates with a limited local view of incoming traffic. To provide a more predictive 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. A series of real-time traffic signal control policies were then defined that use this higher level representation of traffic flows as look-ahead guidance. We compared our method, PBSS to fixed timing plans and two recently proposed adaptive strategies, one based exclusively on queue-clearing, and another based on a simple measure of traffic density on incoming links. Two traffic networks with dynamic vehicle flows were considered: an arterial road with five intersections and a grid with six intersections. For both networks, 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"—without the need for explicit communication between intersections.

There are several aspects of the proposed self-scheduling model that warrant further investigation. One issue concerns the development of more-informed control policies. The policies investigated in this paper are applied in fixed order, and decisions are based simply on the detected presence of an oncoming platoon. The opportunity exists for designing of more sophisticated methods that search in the aggregate representation space and explicitly evaluate tradeoffs in the case of conflicting platoons. Another idea is to incorporate algorithm portfolio techniques such as [22] to manage application of a suite of scheduling policies on-line.

A second direction for future research is the use of peer-to-peer communi-

cation between neighboring agents to extend the local look-ahead horizon. The platoon-based scheduling policies described in this paper might not be directly applicable in road networks with very short road segments. However, by allowing detected flow components to be communicated from upstream neighbors, a given self-scheduling agent may be able to construct the necessary local view to continue to operate independently and without explicit coordination. It might also be useful to investigate cooperation with network-level coordination mechanisms.

References

[1] G. Balan and S. Luke. History-based traffic control. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 616–621, Hokkaido, Japan, 2006.

[2] A. L. C. Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342–375, 2009.

[3] C. Cai, C. Wong, and B. Heydecker. Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies*, 17(5):456–474, 2009.

[4] P. Chaporkar and S. Sarkar. Stable scheduling policies for maximizing throughput in generalized constrained queueing systems. *IEEE Transactions on Automatic Control*, 53(8):1913–1931, 2008.

[5] S. Cheng, M. Epelman, and R. Smith. CoSIGN: A parallel algorithm for coordinated traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):551–564, 2006.

[6] D. de Oliveira, A. Bazzan, and V. Lesser. Using cooperative mediation to coordinate traffic lights: A case study. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 463–470, Utrecht, Netherlands, 2005.

[7] L. B. de Oliveira and E. Camponogara. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(1):120–139, 2010.

[8] C. Gershenson. Self-organizing traffic lights. *Complex Systems*, 16(1):29–53, 2005.

[9] G. Gigerenzer and D. G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological review*, 650–669(4):103, 1996.

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

[11] P. Mirchandani and L. Head. A real-time traffic signal control system: Architecture, algorithms, and analysis. *Transportation Research Part C-Emerging Technologies*, 9(6):415–432, 2001.

[12] C. Papadimitriou and J. Tsitsiklis. The complexity of optimal queuing network control. *Mathematics of Operations Research*, 24(2):293–305, 1999.

[13] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

[14] H. Prothmann, J. Branke, H. Schmeck, S. Tomforde, F. Rochner, J. Hahner, and C. Muller-Schloer. Organic traffic light control for urban road networks. *International Journal of Autonomous and Adaptive Communications Systems*, 3(2):203–225, 2009.

[15] S. Richter, D. Aberdeen, and J. Yu. Natural actor-critic for road traffic optimisation. In *Advances in Neural Information Processing Systems*, pages 1169–1176, Vancouver, BC, Canada, 2006.

[16] D. Robertson and R. Bretherton. Optimizing networks of traffic signals in real time - the SCOOT method. *IEEE Transactions on Vehicular Technology*, 40(1):11–15, 1991.

[17] M. Selinger and L. Schmidt. Adaptive traffic control systems in the United States. Technical report, HDR Engineering, Inc., 2009.

[18] 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*, (2035):69–80, 2007.

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

[20] H. A. Simon. Rational choice and the structure of the environment. *Psychological Review*, 63(2):129–138, 1956.

[21] S. F. Smith. Is scheduling a solved problem? In *MultiDisciplinary Scheduling: Theory and Applications*, pages 3–17. Springer, 2005.

[22] M. Streeter, D. Golovin, and S. F. Smith. Combining multiple heuristics online. In *National Conference on Artificial intelligence*, pages 1197–1203, Vancouver, BC, Canada, 2007.

[23] Texas Transportation Institute. 2009 annual urban mobility report. Technical report, TTI, Texas A&M University System, 2009.

[24] M. Wiering. Multi-agent reinforcement learning for traffic light control. In *International Conference on Machine Learning*, pages 1151–1158, Stanford, CA, 2000.