Transportation Research Part C 24 (2012) 168–189

Contents lists available at SciVerse ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Schedule-driven intersection control

Xiao-Feng Xie [a,*], Stephen F. Smith [a], Liang Lu [b], Gregory J. Barlow [a]

[a] The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, United States
[b] Department of Industrial Engineering and Logistics Management, The Hong Kong University of Science and Technology, Kowloon, Hong Kong

A R T I C L E   I N F O

Article history:
Received 22 July 2011
Received in revised form 16 March 2012
Accepted 16 March 2012

Keywords:
Adaptive traffic signal control
Model-based intersection optimization
Real-time systems
Urban traffic control
Intelligent transportation systems

A B S T R A C T

Model-based intersection optimization strategies have been widely investigated for distributed traffic signal control in road networks. Due to the form of "black-box" optimization that is typically assumed, a basic challenge faced by these strategies is the combinatorial nature of the problem that must be solved. The underlying state space is exponential in the number of time steps in the look-ahead optimization horizon at a given time resolution. In this paper, we present a schedule-driven intersection control strategy, called SchIC, which addresses this challenge by exploiting the structural information in non-uniformly distributed traffic flow. Central to our method is an alternative formulation of intersection control optimization as a scheduling problem, which effectively reduces the state space through use of an aggregate representation on traffic flow data in the prediction horizon. A forward recursive algorithm is proposed for solving the scheduling problem, which makes use of a dominance condition to efficiently eliminate most states at early stages. SchIC thus achieves near optimal solutions with a polynomial complexity in the prediction horizon, and is insensitive to the granularity of time resolution that is assumed. The performance of SchIC with respect to both intersection control and implicit coordination between intersections is evaluated empirically on two ideal scenarios and a real-world urban traffic network. Some characteristics and possible real-world extensions of SchIC are also discussed.

1. Introduction

Traffic signal control in road networks is an important practical problem due to substantially increasing delay and fuel cost caused by traffic congestion (Schrank et al., 2011). 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 offer the most potential. However, the adaptive control problem is challenging. On one hand, the combined number of signal control choices and traffic conditions is huge for even an individual intersection (Papageorgiou et al., 2003; Shelby, 2004) and grows exponentially with the size of the traffic signal network (Papadimitriou and Tsitsiklis, 1999). On the other, the non-linear dynamics in a switched flow system (Chase et al., 1993) and unpredictable driving behaviors make reliable prediction possible only over a limited time horizon and force continual updates of signal timings to provide near optimal results.

Classical signal control systems assume a cyclic operation of traffic lights, where each light moves through its sequence of phases with calculated *split* parameters in a *cycle* that is *offset* from those of its neighbors. In pre-timed control systems, fixed signal timing plans for the intersections in a traffic network can be built using off-line optimization tools, such as TRANSYT (Robertson, 1969), SYNCHRO (Husch and Albeck, 2003), and VISGAOST (Stevanovic et al., 2008), based on historical traffic

* Corresponding author. Tel.: +1 4122687358.
E-mail addresses: xfxie@cs.cmu.edu (X.-F. Xie), sfs@cs.cmu.edu (S.F. Smith), llxad@ust.hk (L. Lu), gjb@cs.cmu.edu (G.J. Barlow).

0968-090X/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.trc.2012.03.004

flow data, for specific time periods (typically on a time-of-day basis). These methods provide macroscopic understanding of traffic flow patterns that have a degree of stability over time. Off-line optimized plans also sometimes serve as a guideline, as a baseline, and/or as contingency plans in more advanced traffic control systems, e.g., SCOOT (Robertson and Bretherton, 1991), and ACS-Lite (Luyanda et al., 2003).

Some traffic-responsive cyclic approaches utilize on-line traffic data. The common cycle length of a given traffic network (or sub-network) might be dynamically adjusted to meet the traffic demand (Sims and Dobinson, 1980; Robertson and Bretherton, 1991). The offsets between intersections might be calculated by using statistical flow profiles (Luyanda et al., 2003) or might be incrementally adjusted (Robertson and Bretherton, 1991) to facilitate good progression for higher traffic flow links. Phase splits might be locally adjusted (Robertson and Bretherton, 1991), or calculated by balancing degree of saturation (Sims and Dobinson, 1980) or phase utilization (Luyanda et al., 2003). Normally, it is difficult for these methods to exploit the significance of short-term (e.g., second-by-second (Wu et al., 2010)) variability of traffic, due to the rather strong restriction imposed by reliance on parametric timing plans.

Many other traffic signal control systems have been built in a bottom-up way (Mirchandani and Head, 2001; Gartner et al., 2002; Papageorgiou et al., 2003). In these systems, intersection control plays a fundamental role in adapting to prevailing dynamic traffic flow data in real time. Each intersection decides independently when to switch among its sequence of phases in a way that enforces basic safety and fairness constraints, rather than being confined to parametric timing plans.

Some reactive intersection control strategies, e.g., vehicle-actuated (VA) logic (Dunne and Potts, 1964; Viti and van Zuylen, 2010) and other self-controlled strategies (Lämmer and Helbing, 2008), make decisions quickly based on simple clues in traffic flow, e.g., a critical interval between vehicles (Dunne and Potts, 1964; Viti and van Zuylen, 2010) and an anticipated queue of vehicles (Lämmer and Helbing, 2008). These methods can obtain reasonably good performance for an isolated intersection (Lämmer and Helbing, 2008). However, due to their myopic nature, these methods are susceptible to suboptimal decisions.

To enable more optimal decisions, model-based intersection control methods (Papageorgiou et al., 2003) have been widely employed. Well-known examples include DYPIC (Robertson and Bretherton, 1974), OPAC (Gartner et al., 2002), PRODYN (Henry et al., 1983), COP (Sen and Head, 1997), ALLONS-D (Porche and Lafortune, 1999), ADPAS (Kim et al., 2005), and CRONOS (Boillot et al., 2006). These methods attempt to produce a globally optimized solution over a specified *optimization horizon*. The underlying state space is formed by dividing this horizon into discrete intervals based on a fixed *time resolution*, and all temporal values are rounded into numbers of time steps. Assuming that the control model is sufficiently accurate to represent the real problem, this space is then searched via an optimization process. To cope with the fact that a reliable *prediction horizon* can be quite limited, these methods typically employ the rolling horizon scheme (Bell, 1992; Newell, 1998).

With a strong optimization capability, isolated intersection control strategies can provide a strong basis for *implicit coordination* between neighbor intersections (Porche and Lafortune, 1999). These strategies can also be integrated into network-wide control systems, e.g., RHODES (Mirchandani and Head, 2001) and RT-TRACS (Gartner et al., 2002), where coordination is explicitly strengthened, either by applying additional signal control guidance from neighboring intersections that incorporates non-local impact or by extending the prediction horizon with flow information from neighboring intersections.

For existing model-based control methods, the basic challenge is computational efficiency, as the underlying state space is exponential in the number of time steps in the optimization horizon (Papageorgiou et al., 2003). Exact methods based on exhaustive search and branch-and-bound (Robertson and Bretherton, 1974; Porche and Lafortune, 1999; Shelby, 2004) are too time-consuming for on-line use in most realistic settings. ALLONS-D, for example, has been shown to be intractable for a prediction horizon of just 15 time steps (Shelby, 2004). To achieve real-time tractability, existing model-based methods often incorporate simple space reduction schemes (e.g., coarser time resolution (Shelby, 2004; Porche and Lafortune, 1999), shorter optimization horizon (Robertson and Bretherton, 1974; Henry et al., 1983; Sen and Head, 1997), smaller number of phase switches (Gartner et al., 2002)), and/or more approximate solving methods (e.g., approximate state elimination (Henry et al., 1983; Sen and Head, 1997), value function approximation (Cai et al., 2009), and heuristic search (Boillot et al., 2006)), both of which can significantly affect solution quality.

In this paper, we focus on the design of a Schedule-driven Intersection Control strategy (SchIC) that can efficiently produce (near) optimal solutions in real time. SchIC achieves efficient search space reduction and state elimination by exploiting structural flow information in the prediction horizon. The intersection control problem is formulated as a scheduling problem, based on an aggregate representation on flow data, where the non-uniformly distributed flow information is used to form a reduced state space that retains promising solutions. Compared to existing single-machine scheduling problems (Koulamas, 2010; Du and Leung, 1990), our scheduling model incorporates some implicit features of the traffic control problem. An optimization procedure is then proposed to solving the scheduling model based on an efficient elimination criterion that reduces the number of state updates significantly. The complexity of the resulting SchIC procedure is polynomial in the prediction horizon, and hence much less dependent on choice of temporal granularity. The capability of SchIC for intersection optimization and implicit coordination between intersections is evaluated on two ideal scenarios and a real-world traffic network. Possible extensions of SchIC in more complex real-world applications, including coordinated network-wide control, are also discussed.

The remainder of the paper is organized as follows. In Section 2, we formulate the intersection traffic control problem in a way that is compatible with existing methods and allows comparison to SchIC. In Section 3, we describe the details of the SchIC algorithm. In Section 4 we present experimental results that indicate the performance of the proposed approach. Next,

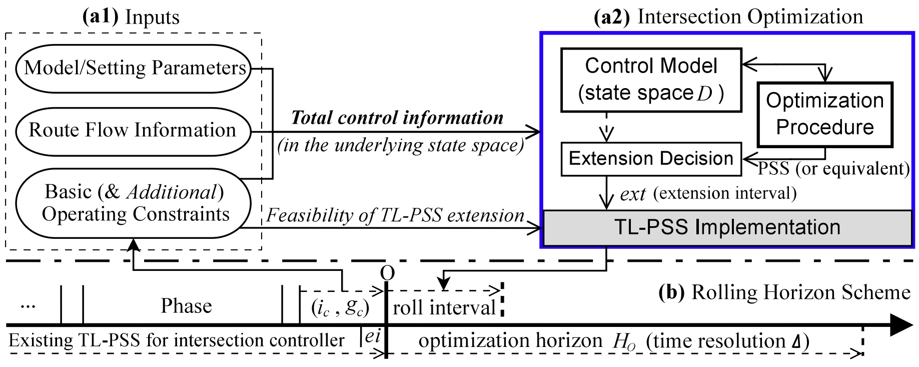
(a1) Inputs
Model/Setting Parameters
Route Flow Information
Basic (& Additional) Operating Constraints
Total control information
(in the underlying state space)
Feasibility of TL-PSS extension
(a2) Intersection Optimization
Control Model (state space D)
Optimization Procedure
Extension Decision
PSS (or equivalent)
ext (extension interval)
TL-PSS Implementation
O
Phase
(i_c, g_c)
roll interval
(b) Rolling Horizon Scheme
Existing TL-PSS for intersection controller
ei
optimization horizon H_O (time resolution Δ)

Fig. 1. Intersection traffic control (a1 and a2) based on a rolling horizon scheme (b).

in Section 5, we discuss some characteristics and possible extensions of SchIC. Finally, in Section 6 we summarize and indicate directions for future research.

2. Intersection traffic control

We consider a signalized intersection with a set of entry and exit approaches, in which each approach has a fixed length and a set of lanes. The fundamental mapping between traffic movements and light signals is defined by the *phase design* containing a set of phases I, in which each *phase index* $i \in [1, |I|]$ corresponds to the right-of-way for a route i. Each *route* contains a set of non-conflicting movements that allow safe passage of vehicles through the intersection, in which each *movement* defines the traffic flow from (some number of the departure lanes of) an entry approach to an exit approach. If one route is allocated the right-of-way (i.e., green light), the traffic light must be red for all other routes. In addition, we assume that all routes are non-overlapping, i.e., do not share any movements.

The *signal timing specification* contains a set of operating constraints on the phase switching process to ensure safety and fairness for all vehicles in the traffic flow:

(1) Phase indices switch in cycles, i.e., the next phase of i is $next(i) = (i + 1)$ modulo $(|I|)$, since a fixed phase sequence is preferred by many traffic engineers (Mirchandani and Head, 2001).
(2) Each phase i has a variable duration between $\left[G_{min}^{(i)}, G_{max}^{(i)}\right]$, in which $G_{min}^{(i)}$ and $G_{max}^{(i)}$ are respectively the *minimum* and *maximum green times* for phase i.
(3) Each switching process from phase i to $next(i)$ requires a fixed *intergreen time* $Y^{(i)}$ (or *effective clearance interval* (Sen and Head, 1997)) during which no vehicles can leave from the intersection.[1]

Each actual *phase* is defined as (i, g), or $g^{(i)}$, in which i is the phase index, g is a variable phase duration. A *phase switching sequence* (PSS) is defined as a sequence of phases, given an initial phase condition (i_c, g_c) at its start time, where i_c is the current green phase index, and g_c is the time that the current phase i_c has been green. Based on the start time, the actual operation period of all phases in a PSS can be calculated by adding their phase durations and corresponding intergreen times between phases. The structure of a PSS allows it to be easily expanded by increasing g_c for the current phase or adding the next phase.

The execution of the traffic light follows a specific PSS, called TL-PSS, that satisfies the signal timing specification over time. For intersection traffic signal control, the objective is to implement a complete TL-PSS that minimizes the cumulative delay (as in Sen and Head (1997), Shelby (2004), and Porche and Lafortune (1999)) of all vehicles during the entire process of approaching and passing through the intersection.

2.1. Basic framework

We adopt a basic framework that is compatible with existing model-based control methods (Mirchandani and Head, 2001; Gartner et al., 2002; Papadimitriou and Tsitsiklis, 1999), in order to provide the right context for discussing our method.

As shown in Fig. 1, intersection control is realized by iteratively extending the existing TL-PSS before each of its finish times, based on the rolling horizon scheme (b) (Bell, 1992; Newell, 1998), to utilize flow information in a limited *prediction horizon* (H_P). Given each finish time of the existing TL-PSS (e.g., O) as the origin point, the total "roll step" of obtaining the

[1] In real life, an intergreen time might contain a part of "yellow" and "all-red" intervals (Sen and Head, 1997), during which vehicles have sufficiently low probability to leave from the intersection.

inputs (a1) and running the intersection optimization strategy (a2) is performed during a short *execution interval* (*ei*) before the finish time, in order to extend the existing TL-PSS for a *roll interval*. The only requirement is that *ei* is smaller than each *roll interval*.

2.1.1. Inputs

For intersection optimization, the input information includes basic (and additional) *operating constraints*, *route flow information*, and related setting parameters.

The basic operating constraints include the signal timing specification and the current phase condition (i_c, g_c), i.e., the last phase of the existing TL-PSS. These constraints are used to ensure feasibility of TL-PSS extension. If necessary, additional operating constraints might be added to incorporate knowledge that is learned/inferred from local information and coordination requirements from neighbors (Mirchandani and Head, 2001). In practical usages, the actual operating constraints in total control information might be a mix of both types.

The route flow information contains the queue size $q^{(i)}$ and *temporal arrival distribution* (Mirchandani and Head, 2001) of vehicles between $\left[0, H_P^{(i)}\right]$ on each route *i*, given the origin time point is located at the stop line of the intersection. The maximum *prediction horizon* is $H_P = \max_{i=1}^{|I|} H_P^{(i)}$.

Setting parameters are constant in each intersection optimization process. Signal control is performed in an *optimization horizon* (H_O), and all temporal values are rounded into numbers of time steps by dividing a fixed *time resolution* (Δ) (Robertson and Bretherton, 1974). Typical model parameters include the *start-up lost time* and *saturation flow rate* on each route (Sharma et al., 2007; Shelby, 2004; Sen and Head, 1997), which can be estimated from historical data (Sharma et al., 2007; Mirchandani and Head, 2001). In addition, there are a few algorithm parameters.

2.1.2. Underlying state space

Given a time resolution Δ, the underlying state space Ω can be described by a decision tree (Robertson and Bretherton, 1974; Porche and Lafortune, 1999; Shelby, 2004). Each PSS is built from time 0 (the root node), and the *state* at its finish time is a partial solution with an objective value (cumulative delay). At each stage of the search, successor states are generated by making decisions to extend the current PSS by one step of Δ, which might belong either to a phase or an intergreen time. For any given state, the corresponding PSS can be assembled by collecting the decisions previously made along the path back to the root node.

Each state is called a *full-clearance* state if it is able to clear all vehicles in the prediction horizon. Each full-clearance state is a natural *leaf state*, since further expansion will not change the objective value. For the complete decision tree, the optimal solution is the one with the minimal cumulative delay among all leaf states, and its finish time is called $H_O^{\#}$. Note that $H_O^{\#}$ might be much longer than H_P, especially for saturated flow conditions.

Without loss of optimality, a sufficiently long optimization horizon H_O $(H_O \geqslant H_O^{\#})$ can be used to limit the finish time of states. A basic challenge is that the size of the state space Ω increases exponentially with $|H_O| = H_O/\Delta$, the number of time steps in H_O (Papageorgiou et al., 2003).

The space Ω might not be implemented directly, but it nonetheless provides a common context for discussing the solution quality and efficiency of different optimization strategies. Efficiency mainly depends on the number of state updates (i.e., visited states) in Ω, and on the average time required for each state update. For a rational optimization, each (near) optimal solution should either be a (near) optimal full-clearance PSS in Ω or part of one.

2.1.3. Intersection optimization

The intersection optimization framework contains a *control model*, an *optimization procedure*, an *extension decision*, and a static component for TL-PSS implementation.

The control model has two basic functions: (1) formation of an actual *state space* ($\mathcal{D}$), which is a subspace of the underlying state space Ω and (2) transition and evaluation of the states in $\mathcal{D}$. The optimization procedure then tries to find a (near) optimal state in $\mathcal{D}$ through some search/optimization strategies. For intersection optimization, the formation of the state space $\mathcal{D}$ (i.e., state space reduction) and the optimization procedure play the *solving role* that is critical to efficient and effective solution generation. The extension decision outputs a time duration *ext* $\geqslant 0$ based on the solution obtained by the optimization procedure (normally the first step or phase).

Based on the basic operating constraints, TL-PSS is implemented in a roll interval by using a static repair rule (Porche and Lafortune, 1999; Kim et al., 2005): If *ext* > 0, the current phase is extended to $g_c = \min\left(g_c + ext, G_{max}^{(i_c)}\right)$. If $g_c \equiv G_{max}^{(i_c)}$ or *ext* $\equiv 0$, the current phase is terminated, and a new phase $\left(i_n, G_{min}^{(i_n)}\right)$ is added, in which $i_n = next(i_c)$, after the intergreen time $Y^{(i_c)}$.

3. Schedule-driven Intersection Control (SchIC)

SchIC is a realization of the above intersection optimization framework that efficiently achieves (near) optimal solutions. The novelty of SchIC stems from its effective utilization of route flow information to strengthen the solving efficiency.

The SchIC algorithm proceeds as follows. First, the route flow information is preprocessed into an aggregate form that captures structural information in the traffic flow. Second, a scheduling model for traffic control is formulated, in which

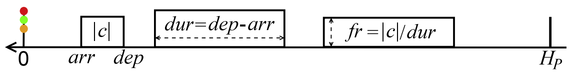

Fig. 2. Example of clusters on a route, each can be defined by three of the five attributes.

the state space ($\mathcal{D}$) (a much smaller subspace of Ω) is formed by using the clusters in the aggregate form. Third, for the optimization procedure, an efficient elimination criterion is proposed, which is able to reduce the number of state updates without loss of optimality, by utilizing an inherent property in the scheduling model. Finally, the extension decision is implemented.

For the actual operation constraints, we adopt a common relaxation (Sen and Head, 1997; Kim et al., 2005; Cai et al., 2009) of the basic operation constraints in which the maximum green times and hence g_c are not included. This relaxation is based on an assumption that phase durations should rarely extend to well-defined maximum green times. Note that the constraints of maximum green times are always ensured by the static rule in the TL-PSS implementation (Section 2.1.3). For simplicity, we do not consider any additional operating constraints in this paper.

Basic traffic models are used (Sharma et al., 2007; Mirchandani and Head, 2001): On each road, all vehicles are moving at constant *free-flow* speed without any stopping; On each route, a queue of vehicles is discharged in a green phase at the *saturation flow rate* (*sfr*) after the *start-up lost time* (*slt*).

3.1. Aggregate flow representation

The aggregate flow representation is defined by aggregating vehicles on routes into sequences of clusters. More precisely, each $C^{(i)}$ is a cluster sequence on route *i*, which is transformed from the traditional route flow information. Because the transformation process on each route is independent, the route index is omitted in this section.

The cluster sequence *C* associated with a given route represents the non-uniformly distributed traffic flow within the current prediction horizon (H_P). Each *C* contains an ordered sequence of *clusters*, $(c^{(1)}, \ldots, c^{(|C|)})$, where $|C|$ is the number of clusters in *C*. Each *cluster c* has five attributes, $(|c|, arr, dep, dur, fr)$, in which $|c|$ is the number of vehicles in *c*, *arr* (*dep*) gives the expected *arrival time* (*departure time*) in reference to the stop line of the intersection respectively for the first (last) vehicle in *c*, *dur* is the *duration* between *dep* and *arr*, and *fr* is the average *flow rate* when *c* is serviced (by assuming that vehicles are uniformly distributed within each cluster). Clusters in *C* are ordered by increasing *arr* values. In principle, three of these five attributes are enough to define *c* based on two equations, i.e., $dur = dep - arr$ and $fr = |c|/dur$. As illustrated in Fig. 2, each cluster is a block between $[arr, dep]$, and the width, height and area are *dur*, *fr*, and $|c|$, respectively.

The basic intuition is that cumulative delay is only associated with those vehicles (in clusters) that are not serviced in time. For a cluster with a high *fr* value, it is often better serviced in one phase to avoid a high cumulative delay on the residual queue, due to rather long intergreen and start-up lost times that are wasted in the phase switching process.

On each route, the basic aggregate form of the cluster sequence *C* on the traditional flow information (Shelby, 2004; Sen and Head, 1997) is obtained as follows. If the queue size $q > 0$, it is transformed into a *queue cluster* c_q, which has $|c_q| = q$, $arr(c_q) = 0$, and $fr(c_q) = sfr$, as the first cluster in *C*. Here *sfr* is the *saturation flow rate* (Shelby, 2004; Sen and Head, 1997; Sharma et al., 2007). The temporal arrival distribution of vehicles in the prediction horizon H_P is divided into time segments with a fixed sampling interval *samp* for detection, and $a^{(h)}$ is the number of vehicles arriving in the *h*th segment, for $h = 1$ to $H_P/samp$. For the *h*th segment, if $a^{(h)} > 0$, it is transformed into an *arriving cluster c*, which has $|c| = a^{(h)}$, $dep(c) = h \cdot samp$, and $dur(c) = samp$, and is stored into *C*. No rounding error will be introduced if $samp/\Delta$ is an integer, as is considered in this paper.

This flow representation can be further aggregated through use of two techniques, which serve to reduce the problem size while retaining the structural information in the traffic flow:

1. *Threshold-based clustering* – This technique captures a feature used by vehicle-actuated logic (VA) (Papageorgiou et al., 2003; Viti and van Zuylen, 2010), where vehicles are continuously serviced if there are within a critical interval. Aggregate clusters are formed by *merging* any two arriving clusters when the time gap between them is within a specified threshold $thc \geqslant 0$. In this case, two clusters $c_{(1)}$ and $c_{(2)}$ are *merged* into one cluster $c_{(0)}$, which has $arr(c_{(0)}) = \min(arr(c_{(1)}), arr(c_{(2)}))$, $dep(c_{(0)}) = \max(dep(c_{(1)}), dep(c_{(2)}))$, and $|c_{(0)}| = |c_{(1)}| + |c_{(2)}|$.
2. *Anticipated queue clustering* – On each route, an anticipated queue contains the number of vehicles that are presently or in the future will join the queue before the existing queue clears at the intersection (Lämmer and Helbing, 2008). The anticipated queue is built by extending the initial queue cluster c_q, such that both *arr* and *fr* values always remain unchanged, whereas $|c_q|$ might be increased if any vehicles join into the queue. Both *dep* and *dur* values of c_q will increase accordingly if $|c_q|$ increases.
   In more detail, the clusters in *C* are examined one by one. For the *j*th cluster $c^{(j)}$, if $arr(c^{(j)}) \leqslant dep(c_q)$, the anticipated queue is expanded by the following process:
   (1) If $dep(c^{(j)}) \leqslant dep(c_q)$ or $fr(c^{(j)}) \geqslant fr(c_q)$, $c^{(j)}$ totally joins into c_q, i.e., $|c_q| = |c_q| + |c^{(j)}|$, and $c^{(j)}$ is removed from *C*.
   (2) Otherwise, a part or all of $c^{(j)}$ might join into c_q, and $dur(c_q)$ is extended for

$$\delta dur = (dep(c_q) - arr(c^{(j)}))/(1 - fr(c^{(j)})/sfr),$$

based on whether the flow rate of the later part of $c^{(j)}$ remains unchanged:

(a) If $\delta dur \geqslant dur(c^{(j)})$, then $c^{(j)}$ joins into c_q, and $c^{(j)}$ is removed from C.

(b) Otherwise, the earlier part of $c^{(j)}$ joins into c_q, i.e., $|c_q| = |c_q| + |c^{(j)}| \cdot \delta dur/dur(c^{(j)})$, and the later part of $c^{(j)}$, which has $arr = arr(c^{(j)}) + \delta dur$, $dep = dep(c^{(j)})$, and $fr = fr(c^{(j)})$, becomes a new cluster that remains in C.

The aggregation process for an anticipated queue is terminated if there is $arr(c^{(j)}) > dep(c_q)$ or after the choice (2b) is reached.

Proposition 1 can be obtained from the basic aggregate form resulting from the threshold-based clustering. Note that the aggregation of an anticipated queue can further reduce $|C|$.

Proposition 1. *On each route, there is* $|C| \leqslant H_P/(thc + \Delta) \leqslant |H_P|$.

Here $|H_P| = H_P/\Delta$ is the number of time steps in the prediction horizon H_P. The upper bound of $|C|$ is less sensitive to a fine time resolution Δ, if thc is larger than Δ. In practice, the average size of C might be much smaller due to a non-uniform distribution of vehicles.

3.2. Scheduling model

In this section, the traffic control problem is modeled as a single machine scheduling problem, by respectively viewing the intersection as the machine and clusters in the aggregate flow representation of different routes as jobs. For convenience, the jth cluster in $C^{(i)}$ is called $c^{(i,j)}$, where $j \in [1, |C^{(i)}|]$, and $|C^{(i)}|$ is the number of clusters on the route i.

We also make the following two assumptions. Assumption 1 is used for ensuring the usage of aggregate clusters. Assumption 2 can be mostly derived from the constant (free-flow) speed (Henry et al., 1983; Sen and Head, 1997), for those arriving vehicles in the prediction horizon.

Assumption 1 (*Non-divisible*). Each job is non-divisible, which means that all vehicles in one cluster leave the intersection in the same phase.

Assumption 2 (*Precedence*). On each route i, $c^{(i,j)}$ leaves the intersection after $c^{(i,j-1)}$.

Based on Assumption 1, a *schedule* S specifies the order in which all clusters will pass through the intersection one by one. A straightforward representation is to use a sequence of jobs, in which each job is a unique cluster $c^{(i,j)}$. However, this representation is not convenient for constructing a feasible schedule that satisfies the precedence constraints.

Based on the fact that each job is processed in one phase and each phase services the route with the same index, S is instead represented as a sequence of route indices, i.e., $(s^{(1)}, \ldots, s^{(k)}, \ldots, s^{(|S|)})$, in which $|S| = \sum_{i=1}^{|I|} |C^{(i)}|$. At the kth stage ($k \in [1,|S|]$), the schedule lets the kth job, which is the earliest cluster that remains on the route $s^{(k)}$, leave from the intersection at the earliest time. The kth job is obtained in a deterministic way, based on the concept of a partial schedule and the corresponding schedule status.

A *partial schedule* $S^{(k)}$ contains the first k elements of a schedule. For each partial schedule, its *schedule status* is $X = (x^{(1)}, \ldots, x^{(i)}, \ldots, x^{(|I|)})$, where $x^{(i)} \in [0, |C^{(i)}|]$ counts the number of the elements in $S^{(k)}$ that are equal to i. In other words, $x^{(i)}$ indicates that the first $x^{(i)}$ clusters on the route i have been scheduled. The kth job is the cluster $c^{(s^{(k)}, x^{(s^{(k)})})}$.

In addition to ensuring the feasibility of each schedule, a nontrivial usage of this representation is that $X^{(k)}$ and $s^{(k)}$ are state variables that can be directly obtained from $S^{(k)}$.

For the state space $\mathcal{D}$ of the scheduling model, each *state* is associated with a *partial schedule* $S^{(k)}$. For each $S^{(k)}$, the corresponding state variables are defined as a tuple, (X, s, t, d), where $s = s^{(k)}$, t is the *actual finish time* of the kth job (or the *makespan* of $S^{(k)}$), and d is the cumulative delay for all k jobs. Each $S^{(k)}$ can be automatically interpreted as a phase switching sequence (PSS). For the corresponding PSS, t is its finish time, and s is the current phase index at the finish time. Both s and X can be directly obtained from $S^{(k)}$, whereas t and d are calculated. When necessary, a variable is referred to by the stage k at which it occurs by using a superscript "k" (e.g., $s^{(k)}$ and $t^{(k)}$).

For convenience, we define two unique X arrays, i.e., X_{empty} and X_{full}, which have $x^{(i)} = 0$ and $x^{(i)} = |C^{(i)}|$ for $\forall i$, respectively, corresponding to empty and full status, respectively. For any feasible schedule, its schedule status is always X_{full}.

Each schedule can be iteratively constructed. The initial condition is the *empty* schedule $S^{(0)}$ that has $(X, s, t, d) = (X_{empty}, i_c, 0, 0)$, where i_c is the current phase index at the stage 0. At the kth stage ($k \in [1, |S|]$), $S^{(k)}$ is obtained by adding $s^{(k)}$ to $S^{(k-1)}$, and the state variables are updated in Algorithm 1. In Line 1, X is updated by adding one to the route $s^{(k)}$. The kth job is the cluster $c^{(i,j)}$, which has $i = s^{(k)}$, and $j = x^{(i)}$. In Line 2, Algorithm 2 is called for updating t and d. In Line 3, s is updated.

Algorithm 2 gives the details for adding the kth job, i.e., the cluster c. In Line 2, the *MinSwitch* operation returns the minimum time required for switching from $i_{(A)}$ to $i_{(B)}$, i.e.,

$$MinSwitch(i_{(A)}, i_{(B)}) = \sum_{e=1}^{|E|-1} Y^{(i^{(e)})} + \sum_{e=2}^{|E|-1} G_{min}^{(i^{(e)})}, \tag{1}$$

where the path $E = (i^{(1)}, \cdots, i^{(e)}, \ldots, i^{(|E|)})$, with $i^{(1)} = i_{(A)}$, $i^{(|E|)} = i_{(B)}$, and $i^{(e+1)} = next(i^{(e)})$ contains all unique phases to be passed through during the phase switching process. In other words, the minimum green times are applied for any intervening phases.

Algorithm 1. Updating for (X, s, t, d) of $S^{(k)}$

```
Require: (1) $(X, s, t, d)$ of $S^{(k-1)}$; (2) $s^{(k)}$          {$s^{(k)}$ is the element to be added}
1: $i = s^{(k)}$, $x^{(i)} = x^{(i)} + 1$, $j = x^{(i)}$                {Update the scheduling status $X$}
2: $(t,d)$ = Algorithm 2, given $(s, t, d)$ of $S^{(k-1)}$, and $c^{(i,j)}$   {The $k$th job to be added is $c^{(i,j)}$}
3: $s = s^{(k)}$
```

Algorithm 2. Calculation for (t, d) of $S^{(k)}$ (Subroutine)

```
Require: (1) $(s, t, d)$ of $S^{(k-1)}$; (2) $c$                    {The cluster $c$ is the $k$th job to be added}
1: $i = RouteIndex(c) = s^{(k)}$                                    {The cluster $c$ is located on the route $i = s^{(k)}$}
2: $pst = t + MinSwitch(s, i)$                                      {The permitted start time of $c$}
3: $ast = \max(arr(c), pst)$                                        {The actual start time of $c$}
4: if $(pst > arr(c))$ and $(s \neq i)$ then $ast = ast + sult^{(i)}$   {Considers the start-up lost time}
5: $t = ast + dur(c)$                                               {The actual finish time of $c$}
6: $\delta d = |c| \cdot \max(ast - arr(c), 0)$                      {The local cumulative delay of $c$}
7: $d = d + \delta d$                                               {The total cumulative delay of first $k$ jobs}
8: return $(t, d)$ of $S^{(k)}$
```

The results of *MinSwitch* can be retrieved from a static $|I| \times |I|$ matrix with precalculated data, since all Y and G_{min} values are static. Thus, Proposition 2 can be obtained.

Proposition 2. *Algorithm 2 can be executed in constant time.*

The inherent control process from $S^{(k-1)}$ to $S^{(k)}$ is the extension of the corresponding PSS between $[t^{(k-1)}, t^{(k)}]$. If *pst* is the permitted start time for the kth job, as calculated in Line 2, then the PSS between $[t^{(k-1)}, pst]$ is based on the minimum switching process from $s^{(k-1)}$ to $s^{(k)}$, and the phase index between $[pst, t^{(k)}]$ is $s^{(k)}$.

In Line 1, we simply indicate the fact that the route index of the kth job is $s^{(k)}$. In Lines 3 and 4, the actual start time of the kth job is calculated, in which $sult^{(i)}$ is the *start-up lost time* (Sharma et al., 2007) on the route i. For the two conditions in Line 4, $pst > arr(c)$ means that c has been a queue, and $s^{(k-1)} \neq s^{(k)}$ means c is the first cluster for the current phase. Lines 5 and 6 are based on a simple model that the job duration (or processing time) is fixed, and the local cumulative delay (or weighted tardiness (Koulamas, 2010)) is proportional to $|c|$.

The *objective* is to find a full schedule S^* with the minimal cumulative delay.

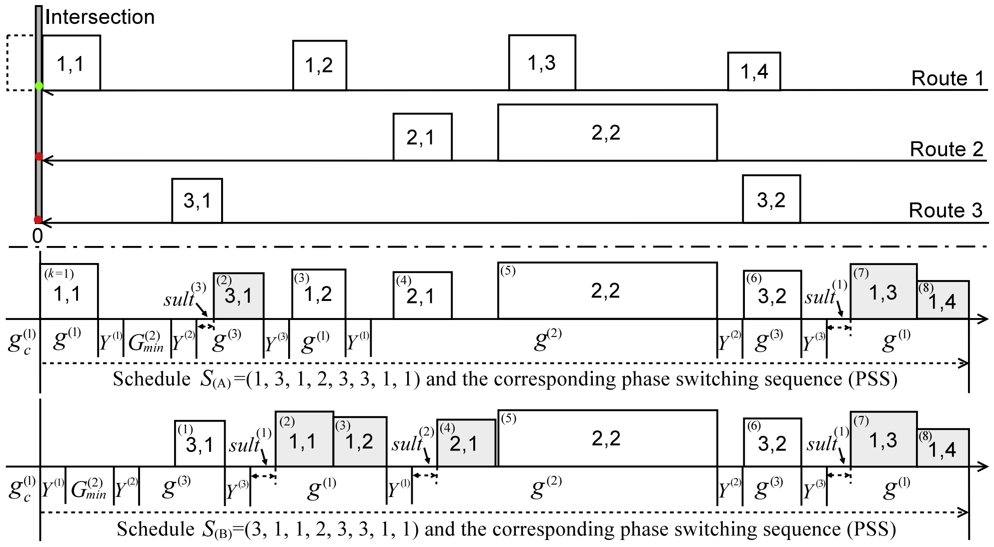
Intersection
1,1
1,2
1,3
1,4
Route 1
2,1
2,2
Route 2
3,1
3,2
Route 3
0
Schedule S(A)=(1, 3, 1, 2, 3, 3, 1, 1) and the corresponding phase switching sequence (PSS)
Schedule S(B)=(3, 1, 1, 2, 3, 3, 1, 1) and the corresponding phase switching sequence (PSS)

Fig. 3. Schedules of jobs (clusters) and the corresponding phase switching sequences.

3.2.1. An example

A simple example is shown in Fig. 3 to illustrate the details of Algorithm 2 and the corresponding PSS construction process. There are four, two, and two clusters on the routes 1, 2, and 3, respectively. The initial phase index is $i_c = 1$. Each label "i, j" means the cluster $c^{(i,j)}$, i.e., the jth cluster on the ith route. We also provide two possible schedules $S_{(A)}$ and $S_{(B)}$ for these clusters. The schedule $S_{(A)}$ is $(1, 3, 1, 2, 2, 3, 1, 1)$. Taking $k = 3$ as an example, we have $s = \underline{1}$, $X = (\underline{2}, 0, 1)$, and the 3rd job is $c^{(1,2)}$. For each corresponding PSS, the phases at different locations in a PSS are different phases, and thus their phase durations might be different, even if there are denoted by the same $g^{(i)}$.

The finish time $t^{(k)}$ at the kth stage is always the actual finish time of the kth job. Some jobs might be in the same phase, e.g., the 4th and 5th jobs, i.e., the clusters $c^{(2,1)}$ and $c^{(2,2)}$.

Several typical cases are shown for the minimum switching process in Eq. (1). For $k = 1$, there are $s^{(0)} = s^{(1)} = 1$, thus $E = (1)$, and $pst^{(1)} = t^{(0)} = 0$. For $k = 2$, $s^{(1)} = 1$ and $s^{(2)} = 3$, thus $E = (1, 2, 3)$, and $\left(Y^{(1)}, G^{(2)}_{min}, Y^{(2)}\right)$ is the PSS between $[t^{(1)}, pst^{(2)}]$. Note that the minimum green time of phase 2 is included, since phase skipping is not allowed. For $k = 3$, $s^{(2)} = 3$ and $s^{(3)} = 1$, thus $E = (3,1)$, and $(Y^{(3)})$ is the PSS between $[t^{(2)}, pst^{(3)}]$.

For the actual start time in Lines 3 and 4 of Algorithm 2, five jobs ($k = 1, 3, 4, 5, 6$) have not become a queue yet, the 8th job is not the first cluster in its phase, and only two jobs ($k = 2, 7$) are further delayed by start-up lost times. All delayed clusters ($k = 2, 7, 8$) contribute to the local cumulative delay that is calculated in Line 6 of Algorithm 2.

The schedule $S_{(B)}$ is $(3, 1, 1, 2, 2, 3, 1, 1)$, which is only different from $S_{(A)}$ in the first two elements. As will be shown in Theorem 1, $S_{(B)}$ can be removed at $k = 3$, since $S^{(3)}_{(A)}$ and $S^{(3)}_{(B)}$ have the same (X, s), and (t, d) of $S^{(3)}_{(B)}$ is dominated by (t,d) of $S^{(3)}_{(A)}$.

3.2.2. Properties

For the scheduling model, the state space $\mathcal{D}$ is a much smaller subspace of the underlying state space Ω. As can easily be seen, the space reduction achieved via the PSS extension at each stage of the scheduling model is significant relative to the single time step Δ that is used to search Ω. In other words, each decision made at each stage of the scheduling model can be seen as a "heuristic leap" over multiple stages in Ω, or equivalently as a bundle of simple heuristics that utilize problem features.

The first heuristic that is incorporated is "do not split a cluster", based on Assumption 1. Each cluster is either present in the atomic form or aggregated as a critical interval (Dunne and Potts, 1964; Viti and van Zuylen, 2010) or an anticipated queue (Lämmer and Helbing, 2008). This aggregate clusters tend to promote a high flow rate, as each aggregation increases the cluster's duration. To split such a cluster is normally not a promising choice due to rather long intergreen times and start-up lost times that are wasted in a phase switching process. Naturally, each stage is defined at the finish time of each job.

Two additional heuristics motivate use of the minimum switching process in Eq. 1. The second heuristic is "do not explore in an idle time period" between the earliest permitted and actual start times of a job. Each idle time period leads to some neighboring PSS alternatives that have the same objective value, and only one representative for them is enough. The third heuristic is "do not intentionally defer a scheduled cluster", i.e., a cluster departs from the intersection at the earliest start time after it is scheduled. Any deferring time duration can only increase the cumulative delay at the intersection.

Given a schedule status X, the order of the jobs on the same route satisfies the precedence constraints in Assumption 2, whereas the order of jobs on different routes is unknown.

Proposition 3. *The number of states with the same X is* $P(X) = \left(\sum_{i=1}^{|I|} x^{(i)}\right)! / \prod_{i=1}^{|I|} (x^{(i)}!)$.

Only full schedules are considered as potential solutions. Based on Proposition 3, The total number of full schedules is $P(X_{full})$. Given the huge number of possible schedules, it might be reasonable to argue that promising schedules in $\mathcal{D}$ have a great possibility to overlap with (near) optimal solutions in the underlying state space Ω.

The schedule status X not only indicates which clusters have left the intersection, but also identifies which clusters have not yet left (the complement of X) on each route. Based on Proposition 3, many states shares the same X. Proposition 4 gives an essential property.

Proposition 4 (*Nondecreasing*). *For Algorithm 1, (t, d) of $S^{(k)}$ are nondecreasing with respect to (t, d) of $S^{(k-1)}$, given all other inputs, i.e., (X, s) of $S^{(k-1)}$ and $s^{(k)}$, are same.*

The proof is straightforward, since both δd and $t^{(k)}$ in Lines 5 and 6 of Algorithm 2 are nondecreasing with $t^{(k-1)}$, and $d^{(k)} = d^{(k-1)} + \delta d$ in Line 7.

Thus, some states in $\mathcal{D}$ can be eliminated at the early stage, as shown in Theorem 1.

Theorem 1. (*Elimination Criterion*). *For two partial schedules $S^{(k)}_{(A)}$ and $S^{(k)}_{(B)}$ with the same (X,s), if (t,d) of $S^{(k)}_{(B)}$* is dominated *by that of $S^{(k)}_{(A)}$, i.e., $\left(t \text{ of } S^{(k)}_{(A)}\right) \leqslant \left(t \text{ of } S^{(k)}_{(B)}\right)$ and $\left(d \text{ of } S^{(k)}_{(A)}\right) \leqslant \left(d \text{ of } S^{(k)}_{(B)}\right)$, then $S^{(k)}_{(B)}$ can be eliminated without loss of optimality.*

Proof. Given the same (X, s), we can consider two *arbitrary* full schedules $S_{(A)}$ and $S_{(B)}$ that are different only in the order of the first $(k - 1)$ elements corresponding to $S^{(k)}_{(A)}$ and $S^{(k)}_{(B)}$, and thus their $X^{(k)}$ and $s^{(k)}$ elements are always the same as k increases to $|S|$. Based on Proposition 4, if (t, d) of $S^{(k)}_{(A)}$ dominates that of $S^{(k)}_{(B)}$, then the dominance relation holds for the $(k + 1)$ th stage. Thus, $(d \text{ of } S^{(|S|)}_{(A)}) \leqslant \left(d \text{ of } S^{(|S|)}_{(B)}\right)$. □

Compared to existing single-machine total weighted tardiness scheduling problems (Koulamas, 2010; Du and Leung, 1990), this scheduling model has the special feature that jobs sharing the same route are subject to precedence constraints on that route. Furthermore, there are two nontrivial properties from the traffic control problem, i.e., the number of routes $|I|$ is small, and the number of time steps in the prediction horizon $|H_P|$ is limited.

3.3. Optimization procedure

In Section 3.2, we have defined the state space $\mathcal{D}$ of possible partial schedules, with the state variables (X, s, t, d) describing each partial schedule. In this section we specify a procedure for obtaining the optimal solution in this space.

The optimization procedure gains its power by efficient state elimination. The actual state grouping at each stage is based on Theorem 1. A forward recursion process is then used for obtaining the optimal schedule. The overall time complexity is obtained from analysis of the upper bound of the group size and the optimization horizon.

3.3.1. State grouping and retrieval of partial schedule

The states in $\mathcal{D}$ are organized into state groups. A *state group* is defined as a tuple (X,s), in which X is the schedule status, and $s \in [1, |I|]$ means that the last job is scheduled on the route s. Each state group contains a table of *value rows*, in which each value row contains four elements, i.e., $(\tilde{t}, \tilde{d}, \tilde{s}, \tilde{y})$, and $|(X, s)| \geqslant 0$ is the actual numbers of value rows.

At each stage k, states are organized into multiple groups, where $\sum_{i=1}^{|I|} x^{(i)} = k$ for each group (X,s). Although the size of a state group can be very large in theory (Proposition 3), many states in the group can be eliminated at a given stage k, based on Theorem 1.

For a given state group (X,s), each row index y (or the yth value row) corresponds to a unique state, i.e., a partial schedule $S^{(k)}$ $\left(k = \sum_{i=1}^{|I|} x^{(i)}\right)$, in which each job element can be obtained by tracking back through the state space by using $(\tilde{s}, \tilde{y})$, as shown in Algorithm 3. For this partial schedule $S^{(k)}$, the corresponding state variables are $(X, s, \tilde{t}, \tilde{d})$, and $\tilde{d}$ is the corresponding cumulative delay, where $(\tilde{t}, \tilde{d})$ are in the yth value row of (X,s).

Algorithm 3. Retrieval of the partial schedule $S^{(k)}$ from the tuple (X, s, y)

```
1: for $l = k$ to 1 do
2:     $s^{(l)} = s$                                                        {The $l$th element of $S^{(k)}$}
3:     $(s, y) = (\tilde{s}, \tilde{y})$ in the $y$th value row of $(X, s)$, and $x^{(s^{(l)})} = x^{(s^{(l)})} - 1$
4: end for
5: return $S^{(k)} = (s^{(1)}, \ldots, s^{(l)}, \ldots, s^{(k)})$          $\left\{|S^{(k)}| = \sum_{i=1}^{|I|} x^{(i)}\right\}$
```

3.3.2. Forward recursion

Algorithm 4 describes the forward recursion process. Initially, only the state group (X_{empty}, i_c), in which i_c is the initial phase index at the stage 0, has one value row $(0, 0, -, -)$. This initial state is corresponding to the empty schedule $S^{(0)}$, which has $s^{(0)} = i_c$, $t^{(0)} = 0$, and $d^{(0)} = 0$. For all other state groups (X, s), their value rows need be calculated.

The forward recursive process starts with $k = 1$, proceeds recursively to $k = k + 1$, and terminates until $k = |S|$. At the kth iteration, the set $\underline{X}^{(k)}$ is collected. Then for $\forall X \in \underline{X}^{(k)}$ and $\forall s \in [1, |I|]$, each state group (X, s) is calculated by Algorithm 5. Here the state group (X, s) means the kth job is added on the route s, thus Line 4 contains a condition $x^{(s)} > 0$, where $x^{(s)}$ is the sth element of X. Using the set $\underline{X}^{(k)}$ is only a naive way for ensuring that all input state groups, i.e., (X_O, s_O) for $\forall s_O \in [1, |I|]$, are available in Algorithm 5.

Algorithm 4. Forward recursion process for calculating all state groups

```
Require: $(X_{empty}, i_c)$ has one value row $(0, 0, -, -)$        {The empty schedule $S^{(0)}$}
1: for $k = 1$ to $|S|$ do
2:     Collects the set $\underline{X}^{(k)} = \left\{X : \sum_{i=1}^{|I|} x^{(i)} \equiv k\right\}$
3:     for $\forall X \in \underline{X}^{(k)}$, $\forall s \in [1, |I|]$ do
4:         if $x^{(s)} > 0$ then Execute Algorithm 5 for calculating the state
       group $(X, s)$
5:     end for
6: end for
7: return The solution $S^{*}$                                    {By tracking back from the state
                                                                   $(X_{full}, \arg\min_{s,y} \tilde{d}^{(X_{full},s,y)})$}
```

Algorithm 5. Calculation of the value rows in the state group (X, s)

```
**Require:** the state groups $(X_O, s_O)$ for $\forall s_O \in [1, |I|]$, where $X_O = X$ with $x^{(s)} = x^{(s)} - 1$
1: $c = c^{(s,x^{(s)})}$                    {The job to be added is the $x^{(s)}$th cluster on the route $s$}
2: for $s_O = 1$ to $|I|$ do
3:   for $y_O = 1$ to $|(X_O, s_O)|$ do
4:     $(t_O, d_O) = (\tilde{t}, \tilde{d})$ in the $(y_O)$th value row of $(X_O, s_O)$
5:     $(t, d)$=Algorithm 2, given $(s_O, t_O, d_O)$, and $c$
6:     if $t \leqslant H_O$ then $StateManager(X, s) \leftarrow (t, d, s_O, y_O)$     {Examines each value row}
7:   end for
8: end for
```

Algorithm 5 collects the value rows in the state group (X, s), in which each value row corresponds to a state. Line 1 gives the new job c to be added. Line 2 refers to the last phase index s_O of the previous state group (X_O, s_O), and Line 3 gives the row index for each previous state. Line 4 then retrieves the previous state variables $(\tilde{t}, \tilde{d})$, and Line 5 is used for updating the state variables of the current state after adding the new job c.

In Line 6, *StateManager* maintains the value rows for each group (X, s). This is the key step where dominated states are eliminated. We define two extreme *StateManager* modes:

- The "*greedy*" mode simply maintains an incumbent value row, and replaces it by each input value row if it has a smaller $\tilde{d}$.
- The "*full*" mode stores a complete set of *non-dominated* value rows. The complete set contains all value rows that are not dominated by any other value rows, based on the dominance comparison on the $(\tilde{t}, \tilde{d})$ values. Note that only one is kept if some value rows have the same $(\tilde{t}, \tilde{d})$ values.

The solution $S^{\tilde{*}}$ can be obtained by the retrieval process in Algorithm 3 that starts from the state $(X_{full}, s^{\tilde{*}}, y^{\tilde{*}})$, in which $s^{\tilde{*}}, y^{\tilde{*}}$ are the results of $\arg\min_{s,y} \tilde{d}^{(X_{full},s,y)}$, and $\tilde{d}^{(X_{full},s,y)}$ is the $\tilde{d}$ value in the yth value row of the state group (X_{full}, s).

3.3.3. State group size

Compared to X, the additional information that is given in (X, s) is the last scheduled cluster $c^{(s,x^{(s)})}$. Thus, the group size $|(X, s)|$ is still of exponential scale prior to any state elimination, based on the result of $P(X)$ that is obtained in Proposition 3.

Proposition 5. *For each valid state group (X, s) with $x^{(s)} > 0$, the total number of possible states is $|(X, s)| = P(X_O)$, in which X_O is equal to X with $x^{(s)} = x^{(s)} - 1$.*

Based on Theorem 1, however, all dominated states in each state group (X, s) can be eliminated without loss of optimality, as in the "full" *StateManager* mode. This leads to a nontrivial proposition that $|(X, s)|$ can be compressed from an exponential size to $|H_O|$.

Proposition 6. *Each $|(X, s)|$ is bounded by $|H_O|$, i.e., the number of time steps in H_O.*

Proof. Note that H_O and all $\tilde{t}$ values are rounded by the time resolution. For the dominance comparison of (X, s), the value rows with the same $\tilde{t}$ value can be compared together to keep the one with the minimal $\tilde{d}$ value. Any value row will be eliminated if it has $\tilde{t} > H_O$. □

The states in $\mathcal{D}$ can be seen as being grouped by using (X, s, t). Each (X, s, t), if exists, is a subgroup of (X, s), and only keeps the element with the minimal $\tilde{d}$ value.

3.3.4. Optimization horizon

Suppose the finish time of the optimal schedule S^* in the state space $\mathcal{D}$ is H_O^*, and the minimum and maximum finish times of all schedules are H_O^{min} and H_O^{max}. In Algorithm 5, only those schedules with makespans shorter than H_O will be candidates. Thus, Algorithm 4 might lose the optimal solution if $H_O < H_O^*$, and even return no solution if $H_O < H_O^{min}$.

The actual finish time of any feasible schedule can be used as H_O to ensure at least one solution. For example, a naive all clearing schedule, $AllClear(nc)$, sets H_O as that the sum of the clearance times of all routes pluses the intergreen times in nc cycles ($nc \geqslant 1$).

In addition, such a horizon is quite likely to be a large enough bound to contain H_O^*, if there is a high correlation between a short finish time and a low cumulative delay.

Finally, H_O^{max} is limited in the scheduling space $\mathcal{D}$, as shown in Proposition 7.

Proposition 7. *In the scheduling search space $\mathcal{D}$, $|H_O^{max}|$ is polynomial in $|H_P|$.*

3.3.5. Complexity

The efficiency of SchIC depends mainly on the number of state updates in the optimization procedure that is described in Algorithm 4.

Proposition 8. *Algorithm 4 has the worse case of* $SG \cdot |I|^2 \cdot \prod_{i=1}^{|I|}(|C^{(i)}|+1)$ *state updates, in which* $SG = \max_{X,s}|(X, s)|$*, and each state update contains Lines 4–6 of Algorithm 5.*

Here $\prod_{i=1}^{|I|}(|C^{(i)}|+1)$ is the number of possible X arrays, i.e., $\sum_{k=1}^{|S|}|\underline{X}^{(k)}|$ in Algorithm 4. SG is the upper bound for Line 3 in Algorithm 5, which is dependent on the choice of *StateManager*. $|I|^2$ is from Line 3 in Algorithm 4 and Line 3 in Algorithm 5. For each state update, the main step in Line 5 of Algorithm 5 is finished in constant time (Proposition 2).

Proposition 9 summarizes the complexity of the "greedy" and "full" modes based on Proposition 8, where the SG values are respectively equal to 1 and no larger than $|H_O|$ (based on Proposition 6), and there is $|C^{(i)}| \leqslant |H_P^{(i)}|$ for $\forall i$ (based on Proposition 1).

Proposition 9. *For Algorithm 4, the "greedy" mode has the worse case of* $|I|^2 \cdot \prod_{i=1}^{|I|}(|H_P^{(i)}|+1)$ *state updates, and the "full" mode has* $|H_O|$ *times more updates than the "greedy" mode.*

The "greedy" mode can work with an unlimited H_O, although it is not guaranteed to be optimal. The "full" mode is optimal in the scheduling space $\mathcal{D}$ if $H_O \geqslant H_O^*$. Note that $|H_O|$ of *AllClear*(nc) and $|H_O^{max}|$ in the space $\mathcal{D}$ are polynomial in $|H_P|$ (Proposition 7).

3.4. Extension decision

After the optimization process, a solution $S^{\circledast}$ is obtained. The complete phase switching sequence (PSS) need not actually be generated, since only the first job is used, if available.

There are two choices for the extension decision:

(1) $ext = 0$, if $|S^{\circledast}| \equiv 0$, or $s^{(1)} \neq i_c$, or $arr(c_1) \geqslant SwitchBack(i_c)$.
(2) $ext = dep(c_1)$, otherwise.

Here $s^{(1)}$, $arr(c_1)$, and $dep(c_1)$ are respectively the route index, the expected start and departure times of the first job $c_1 = c^{(s^{(1)},1)}$ of $S^{\circledast}$. $SwitchBack(i) = MinCycle - G_{min}^{(i)}$ is the minimum time required for the traffic light to return to the phase index i in one cycle.

The condition $|S^{\circledast}| \equiv 0$ simply means that there is no clusters on all routes. The condition $s^{(1)} \neq i_c$ indicates a straightforward phase switching process. The long idle time duration before the first job is explicitly squeezed out, if there is $arr(c_1) \geqslant SwitchBack(i_c)$.

The current phase is only extended to $dep(c_1)$. Afterward, the traffic situation will be re-evaluated in the next roll step when new flow information is available.

4. Experiments

We evaluate the performance of traffic control algorithms in simulation using an open-source microscopic road traffic simulator, Simulation of Urban Mobility (SUMO) (Krajzewicz et al., 2002). SUMO was chosen principally because of its accessibility, and although it is not as sophisticated and full-featured as some other commercial simulation products (e.g., VISSIM, CORISM, AIMSUN, Paramics), SUMO's car following model (by Krauss) is comparable (Brockfeld et al., 2004).

Both solution quality and computational cost are measured. For the measure of solution quality, we report the average *speed*(V_n) and *waiting time* (T_w) per vehicle over a given set of approaches to the intersection. Here V_n is equal to the total travel distance for all vehicles divided by the total *travel time* for all vehicles. If the total travel distance is a constant, the travel time is varying inversely to V_n. The difference between the average travel time and *delay* for all vehicles is a constant. T_w is related to the average delay, but T_w does not include the delay during acceleration/deceleration processes. For the measure of computational cost, we report the *number of roll steps* (N_{rs}) and *total CPU time* (T_{cpu}) per run. The average CPU time per roll step can be calculated as T_{cpu}/N_{rs}. In a given time period, N_{rs} is varying inversely to the average roll interval. A shorter average roll interval leads to a tighter bound on the execution interval and more frequent communication with sensors, although the solution quality might be improved due to a quicker revision of TL-PSS (Cai et al., 2009). The computation was executed by JAVA Runtime Environment 1.6 on 3.4 GHz AMD Phenom II. For each instance, we calculate the mean of 100 independent runs. Both T_w and T_{cpu} are reported in seconds, and V_n is reported in m/s.

Our tests use two ideal scenarios, including an isolated intersection and an artery network, and one real-world scenario that is located in the downtown area of Pittsburgh, PA.

Route flow information is obtained by basically using the hybrid technique described in Sharma et al. (2007), which uses a stop-line detector and an advance detector at a fixed distance (L_{det}) upstream on each entry approach. The sampling interval is $samp = 1$ s. This technique provides a prediction horizon $H_P = L_{det}/v_F$ for the vehicle movements on each road, in which v_F is the *free-flow speed* (Sharma et al., 2007; Mirchandani and Head, 2001). The current queue size is continuously updated according to the difference between the number of expected arrival vehicles and the actually departed vehicles at the stop line (Sharma et al., 2007; Boillot et al., 2006; Mirchandani and Head, 2001). If an entry approach has movements in different phases (e.g., one phase for protected left turn and another for through traffic and right turn), the flow data can be allocated to corresponding routes by applying *turning movement proportions* (Cremer and Keller, 1987; Mirchandani and Head, 2001). If a phase services multiple movements (on the same or different entry approaches), the data in all these traffic movements are simply merged into the corresponding route.

The model parameters, i.e., v_F, *sult*, and *sfr*, can be estimated from historical traffic data (Sharma et al., 2007; Mirchandani and Head, 2001). On each route, $sfr = N_{lane}/shw$, where N_{lane} is the number of lanes, and *shw* is the *saturation headway*, i.e., the average headway between vehicles during saturated flow. In our experiments, v_F on each approach is equal to the speed limit, and there are $sult = 3.5$ s and $shw = 2.5$ s on each route. Turning movement proportions, if required, can be dynamically estimated by some classic methods (Cremer and Keller, 1987; Mirchandani and Head, 2001).

4.1. Traffic control methods

Four SchIC versions are evaluated. The version SchIC$_{F0}$ uses the "full" *StateManager* module, but does not use any aggregation techniques. The version SchIC$_{F1}$ also aggregates the anticipated queue. The version SchIC$_{F2}$ further incorporates the threshold-based clustering. The version SchIC$_{G2}$ uses the "greedy" *StateManager* module and both aggregation techniques. For the threshold-based clustering, $thc = 3$ s.

For the "full" mode, we set H_O as the finish time of *AllClear*(2), the naive all clearing schedule that is defined in Section 3.3.4. For the "greedy" mode, there is no limit on H_O.

We compare SchIC versions to a vehicle-actuated logic (VA) (Dunne and Potts, 1964; Shelby, 2004), and an approximate dynamic programing procedure called Controlled Optimization of Phases (COP) (Sen and Head, 1997). In addition, a fixed coordination plan (FIX), and two time-of-day fixed timing plans obtained by SYNCHRO (Husch and Albeck, 2003) are considered for specific scenarios. All traffic control strategies satisfy the signal timing constraints, including the minimum and maximum green times.

The vehicle-actuated logic (VA) is a reactive traffic control strategy (Dunne and Potts, 1964; Shelby, 2004; Papageorgiou et al., 2003; Viti and van Zuylen, 2010). Under the assumption of a constant vehicular arrival rates, this method is shown to be stable and is usually the optimum for a proper choice of control constants (Dunne and Potts, 1964). In this paper, VA is applied on simple scenarios where each route is an entry approach. The current phase is extended if there is a queue or the observed vehicle headway is not larger than the *critical interval* (Papageorgiou et al., 2003) (or extension unit (Viti and van Zuylen, 2010)). Otherwise, the traffic controller switches to the next phase. In this paper, the critical interval is set as 3 s, the same as in Viti and van Zuylen (2010).

COP is the intersection control algorithm used in RHODES (Mirchandani and Head, 2001). It is one of the quickest algorithms to produce high-quality solutions using a 30-step prediction horizon, as shown in a comparison (Shelby, 2004) to OPAC, ALLONS-D, and PRODYN.

For the comparison, our implementation of COP follows the detailed description in Sen and Head (1997), and we consider the variant with a fixed phase sequence, as used in Shelby (2004) and RHODES.

We tested three COP versions. COP0 and COP1 are only different in the optimization horizon settings: COP0 uses the prediction horizon H_P (as in the original setting (Sen and Head, 1997)), whereas COP1 and COP2 use the same optimization horizon with SchIC$_{F0}$ for fully clearing all queuing and arriving vehicles in the horizon. For the rolling horizon process, both COP0 and COP1 extend to the end of the first phase in the optimal PSS (Mirchandani and Head, 2001). COP2 is a variant of COP1, in which the extension interval is capped with a maximum of 5 s.

The time resolution (Δ) in the optimization horizon is 0.5 s, as used in Cai et al. (2009). This choice enables all the temporal variables to be rounded without significant error.

4.2. Ideal scenarios

We consider two ideal scenarios, i.e., an isolated intersection and an arterial network, for evaluating the essential performance of traffic control strategies, respectively on intersection optimization and implicit coordination between neighbor intersections.

For the two simple scenarios, each road has an identical length (L), and each entry approach is active in one phase. On each road, the speed limit is 10 m/s. Each intersection has two phases. By default, the signal timing constraints are $G_{min} = 5$ s, $G_{max} = 55$ s, and $Y = 5$ s, for all phases of every intersection.

4.2.1. Isolated intersection

The isolated intersection scenario with two entry approaches is used for testing the essential performance of traffic control strategies, thus the prediction horizon is long enough and no network-level effect is included. In the real world, such an isolated intersection might be located on a rural road or a major urban road. For each entry approach, the length (L) is 750 m, and the location of the advance detectors is $L_{det} = 700$ m, which corresponds to a 70-s prediction horizon (L_{det}/v_F) with 140 time steps, for the look-ahead optimization strategies (COP and SchIC). This prediction horizon is slightly shorter than

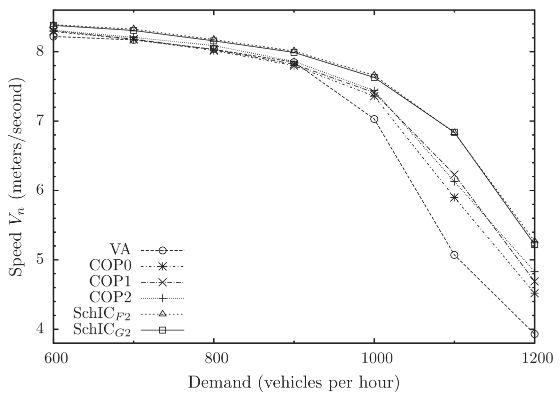

Fig. 4. Average speeds at the isolated intersection with different demands.

the 75-s horizon used in Shelby (2004). In each run, the total simulation period was 1 h, and the traffic demand proportions between two entry approaches were shifted every twenty minutes as follows: 0.3:0.7, 0.5:0.5, and 0.7:0.3.

Figs. 4 and 5 show the two performance metrics, i.e., V_n and T_{cpu}, obtained by VA and the COP and SchIC versions on the isolated intersection with different demands. Table 1 gives the data for the demands $\in \{600, 900, 1200\}$ vehicles per hour.

As shown in Fig. 5 and Table 1, all SchIC versions can reduce the computational cost significantly without sacrificing solution quality. The two aggregation techniques can lead to better performance, which might due to that the further aggregation reduce the significance of errors in *arr* and *dur* values of clusters in each *C*. In the following comparison, SchIC$_{G2}$ is picked as a representative for the SchIC versions, since it is the most efficient SchIC version.

Another observation on the data in Table 1 is that the number of state updates ($N_{ss/rs}$) of SchIC$_{F2}$ is only a little higher than that of SchIC$_{G2}$, which means that the average size of state groups is not large for the "full" mode of the *StateManager* module. In other words, the elimination criterion is able to eliminate most value rows in a state group.

It might be most fair to compare the performance of the optimization procedures of SchIC$_{F0}$ and two COP versions, i.e., COP1 and COP2, since they use the same optimization horizon, and the two aggregation techniques are not applied on the flow representation of SchIC$_{F0}$. As shown in Table 1 and Fig. 5, SchIC$_{F0}$ obtains higher solution quality than COP1 and COP2 with reduced computational cost.

The performance curve of a high-quality traffic control strategy can be divided into two regions, i.e., slowly decreasing in the linear region for low demands and rapidly decreasing in the saturation region for high demands. This is a basic feature in general flows. The performance can be significantly improved in the saturation region, given an effective utilization. As shown in Fig. 4, SchIC$_{G2}$ performs better than all COP versions. Furthermore, the COP and SchIC algorithms obtain higher V_n values than the vehicle-actuated strategy (VA), which might be due to the using of a longer prediction horizon.

As shown in Table 1, the number of roll steps (N_{rs}) of all COP and SchIC versions decreases when the demand increases, except for COP2, which sets a cap on the roll interval. The numbers of state updates per roll step ($N_{ss/rs}$) of all SchIC versions are much lower than that of all COP versions, as shown in the case with 1200 vehicles. SchIC$_{G2}$ requires only 44 state updates, given that the prediction horizon contains 140 time steps, and the optimization horizon is even longer for higher demand.

For an effective control, the maximum green time should be large if the degree of saturation is high, according to Webster's optimal cycle-time formula. Fig. 6 gives the performance of VA, three COP versions, and SchIC$_{G2}$ on the single intersection scenario with different maximum green times. The COP versions and SchIC$_{G2}$ utilize the increase of G_{max} well for enhancing V_n. Among all algorithms, SchIC$_{G2}$ achieves the best performance.

For the three flow parameters, v_F has the most influence on the accuracy of the arrival times of all clusters, whereas *sult* and *shw* only influence the queue clearance time. Fig. 7 investigates the robustness of SchIC$_{G2}$ with different v_F settings when the demands are {600, 900, 1200} vehicles. The performance remains well for the ratios between [0.8, 1.2].

4.2.2. Arterial network

As shown in Fig. 8, the arterial network includes one bottleneck intersection O and four controllable intersections A–D. The intersection O is controlled by a fixed timing plan with the split of 35 s to traffic on the artery in the cycle of 70 s. Only the entry approaches for A–D will be taken into account for performance evaluation. In real world, the intersection O can be seen as the boundary of a control zone. For each intersection, one phase services for a one-way side road, and another for the artery. To study the implicit coordination for traffic control between neighbor intersections, we only consider short prediction horizon intervals: For each entry road, there are $L = 250$ m and $L_{det} = \{50, 100, 150, 200\}$ m, i.e., {5, 10, 15, 20}-s prediction horizon intervals.

The simulation period is 1 h. Incoming traffic is divided among the roads with the proportions shown in Fig. 8, based on the total traffic demand. No turns occur at any intersection except **C**, which has an additional turning movement r_t. For the artery, we set $r_e + r_w = 7/16$. By default, there is $r_w = 0$, which turns the artery into a one-way road. The movement r_s can be

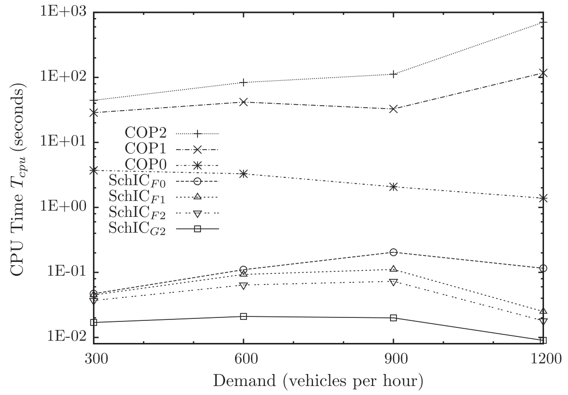

Fig. 5. Average CPU times at the isolated intersection with different demands.

Table 1
Performance at the isolated intersection with different demands.

	600 Vehicles			900 Vehicles			1200 Vehicles			
	V_n	T_{cpu}	N_{rs}	V_n	T_{cpu}	N_{rs}	V_n	T_{cpu}	$N_{ss/rs}$	N_{rs}
VA	8.22	–	–	7.85	–	–	3.93	–	–	–
COP0	8.29	3.287	333	7.80	2.078	190	4.52	1.386	1.88E + 04	118
COP1	8.30	41.82	338	7.82	32.73	197	4.69	117.4	7.87E + 05	120
COP2	8.31	83.50	671	7.86	112.0	659	4.83	704.3	8.37E + 05	677
SchIC$_{F0}$	8.39	0.110	669	8.01	0.204	462	5.05	0.116	7.49E + 02	134
SchIC$_{F1}$	8.38	0.093	557	8.00	0.111	329	5.23	0.025	9.35E + 01	109
SchIC$_{F2}$	8.39	0.064	556	8.01	0.073	331	5.27	0.018	5.67E + 01	110
SchIC$_{G2}$	8.38	0.021	566	7.99	0.020	337	5.22	0.009	4.33E + 01	111

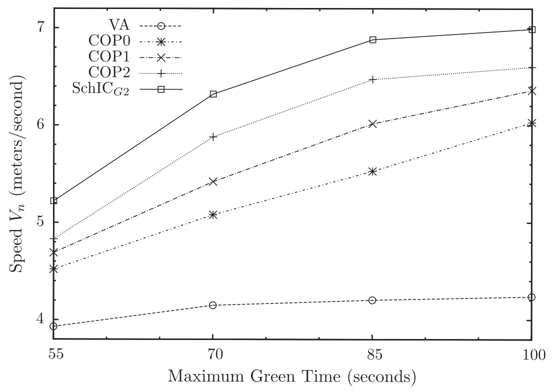

Fig. 6. Average speeds on the intersection with different maximum green times (the demand is 1200 vph).

set to an arbitrary proportion since it does not contribute to the performance evaluation. For simplicity, we set $r_s + r_t = 5/16$. By default, there is $r_t = 0$.

Table 2 reports the results of COP versions and SchIC$_{G2}$ in the control zone, given different demands and L_{det} locations. The arterial waiting time $T_{w,art}$ includes only the four arterial roads leading to intersections A–D, whereas non-bottleneck waiting time $T_{w,nb}$ includes all tw entry approaches leading to these intersections. The performance of these algorithms become better as the prediction horizon is longer. SchIC$_{G2}$ and COP2 obtain the lowest $T_{w,nb}$ values, but COP2 incurs higher computational cost and more roll steps. COP0 cannot optimize when the optimization horizon (5 s for L_{det} = 50 m) is not larger than the intergreen time between two phases.

Table 3 reports the results of FIX, VA, COP versions and SchIC$_{G2}$ for L_{det} = 100 m and different demands. The fixed timing plan (FIX) explicitly coordinates the intersections A–D, which uses 43/17 s splits and 28 s offsets that is chosen by local search by maximizing the average speed as well as satisfying the constraint of $T_{w,art}$ = 0 when r_t = 0.

Traditionally, the coordination between intersections is indicated by achieving "green waves". In this scenario, green waves on the artery can be indicated from a rather small $T_{w,art}$ value. In this sense, VA is not able to achieve coordination,

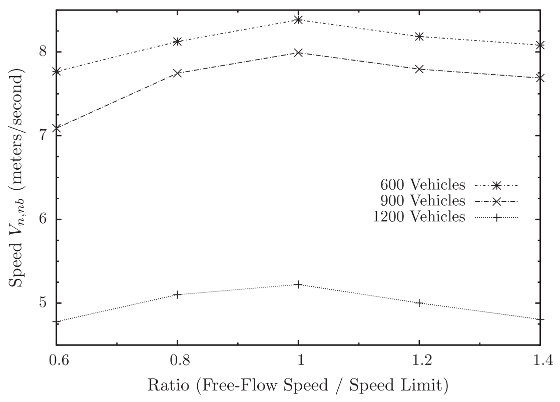

Fig. 7. Average speeds at the isolated intersection with different free-flow speed settings.

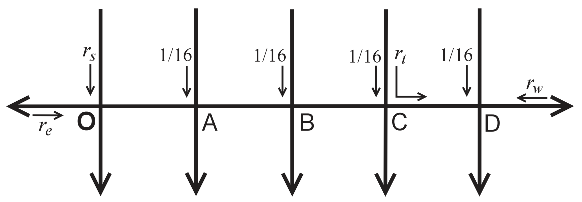

Fig. 8. The arterial network ($r_s + r_t = 5/16$, $r_e + r_w = 7/16$).

Table 2
Performance at the arterial network with different L_{det} locations (1500 vehicles).

	$L_{det} = 50$		$L_{det} = 100$		$L_{det} = 150$		$L_{det} = 200$			
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	T_{cpu}	N_{rs}
COP0	135	91.8	4.43	6.76	5.28	7.79	2.59	7.26	0.112	1330
COP1	2.21	5.70	1.48	4.95	0.76	4.67	0.22	4.79	2.517	1104
COP2	1.77	5.04	1.01	4.24	0.26	4.02	0.06	4.07	6.954	2667
SchIC$_{G2}$	1.78	5.04	0.29	3.71	0.10	3.52	0.11	3.59	0.030	1942

Table 3
Performance at the arterial network with different demands ($L_{det} = 100$ m).

	300 Vehicles		600 Vehicles		900 Vehicles		1200 Vehicles		1500 Vehicles	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
FIX	0.00	5.96	0.00	6.04	0.00	6.15	0.00	6.07	0.00	6.41
VA	6.84	5.33	7.13	5.97	7.15	6.68	7.27	7.52	7.67	8.43
COP0	0.65	1.15	1.17	2.34	1.73	3.45	2.72	4.90	4.43	6.76
COP1	0.39	0.83	0.61	1.89	0.72	2.86	1.07	3.93	1.48	4.95
COP2	0.32	0.74	0.52	1.64	0.58	2.49	0.73	3.40	1.01	4.24
SchIC$_{G2}$	0.26	0.57	0.35	1.23	0.34	2.05	0.27	2.90	0.29	3.71

whereas SchIC$_{G2}$ achieves good coordination ($T_{w,art}$ is 0.29 s) even as the prediction horizon is short (10 s for $L_{det} = 100$ m) and demand is moderately high ($1500 \cdot (7 + 1)/16 = 750$ vph). Moreover, the overall objective should include the performance on side streets, although achieving "green waves" on the artery might be a good sign. FIX has perfect results on $T_{w,art}$, but its $T_{w,nb}$ is rather high. SchIC$_{G2}$ achieves the lowest $T_{w,nb}$ among these strategies.

Fig. 9 shows the average speeds obtained by different control methods at the arterial network with $L_{det} = 100$ m (i.e., a 10-s prediction horizon), given different demands. Here VA is not included in the figure because its solution quality is much worse. For FIX, the average speed only decreases a little when the demand increases for the current scenario. SchIC$_{G2}$ performs better than FIX when demand is less than 1000 vehicles.

The coordination behavior of a fixed strategy might be brittle if there is disturbance on traffic flow. Fig. 10 shows the results for an additional turn movement with different ratios (r_t in Fig. 8) on the side street of **C**, L_{det} is 100 m and the demand

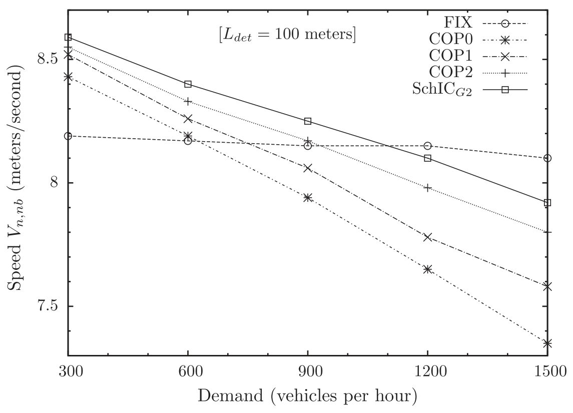

Fig. 9. Average speed at the arterial network with different demands.

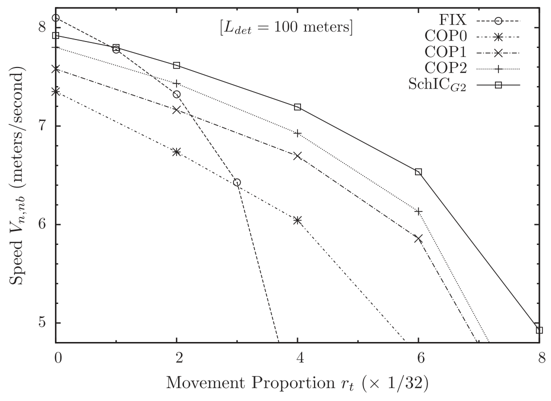

Fig. 10. Average speeds at the arterial network with different r_t values.

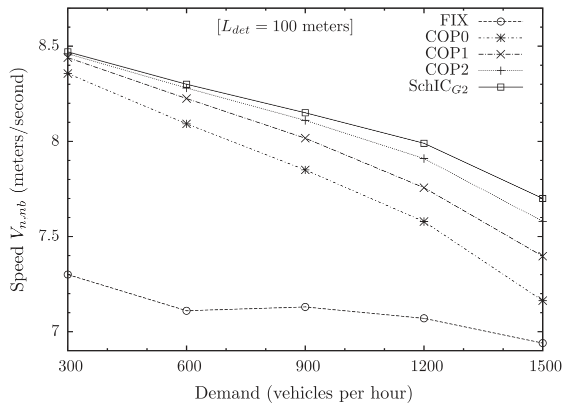

Fig. 11. Average speeds at the arterial network with different demands and varied r_e:r_w proportions.

is 1500 vehicles per hour. Due to a spillback, the performance of FIX drops much quicker than the adaptive strategies. SchIC$_{G2}$ performs better than FIX when r_t > 1/32 (about 47 vehicles).

To understand more about the dynamic behavior of SchIC, we set r_t = 0, but r_e:r_w proportions are changed as 7:0, 4:3, and 1:6, every twenty minutes. Thus the artery contains two-way flows, and the major flow becomes west-bound during the last time period.

Fig. 11 shows the average speeds obtained by different control methods at the arterial network with L_{det} = 100 m. Here the performance of FIX drops significantly. Furthermore, SchIC$_{G2}$ still achieves better performance than the COP versions. The comparison might be unfair for FIX, but major flows do change in the real world. Nevertheless, the results do demonstrate that SchIC can achieve quite good implicit coordination for a two-way artery.

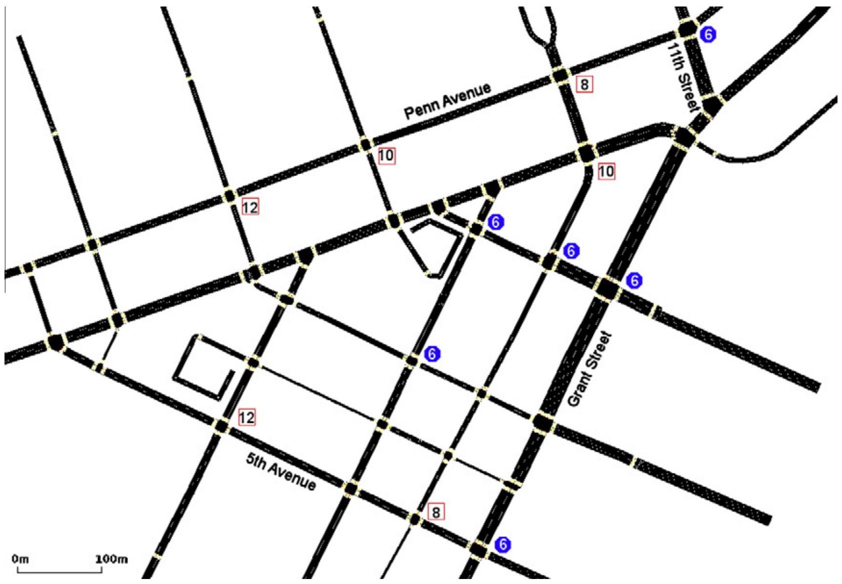

Fig. 12. Downtown Pittsburgh traffic network with 32 intersections.

In summary, a high-quality intersection control strategy can achieve strong implicit coordination between intersections, or at least perform better than a fixed coordination strategy, if the prediction horizon is not too short and the demand is not too high. A nontrivial implication is that a complex traffic network can be decomposed into sub-networks by cutting some links that do not require explicit coordination. Furthermore, a traffic-adaptive strategy might be more robust than a fixed strategy if there are variations in the traffic flow.

4.3. Real-world scenario

Fig. 12 shows the real-world road network of the downtown area of Pittsburgh, PA, which includes the triangle area surrounded by Grant Street, 11th Street, Penn Avenue, and 5th Avenue. Among a total 32 intersections, there are 22 intersections with two phases (of which four have an additional all-red pedestrian clearance interval), 8 intersections with three phases, 1 intersection with one phase plus a pedestrian clearance interval, and 1 intersection with four phases. Most roads allow two-way traffic. The number of lanes on each road ranges from 1 to 3. On each road, the speed limit is 12 m/s.

For the Pittsburgh downtown network, we considered two time-of-day periods corresponding to a off-peak hour and a mid-day hour, called “non-event” and “midday”.[2] These two periods have average vehicle demands of 3933 and 4786 vehicles per hour (vph), respectively. For each time-of-day period, the total simulation time is 4 h.

4.3.1. Results by SYNCHRO

The coordinated signal timing plans currently used to control the Pittsburgh downtown network, including cycle times, splits, and offsets for all intersections, were provided to us by the city and generated using SYNCHRO (Husch and Albeck, 2003), given the traffic signal settings and turning movement counts in various time-of-day periods. SYNCHRO is a widely available, and frequently used commercial software for off-line optimization. As shown in Kamarajugadda and Park (2003), it is difficult to achieve a significant improvement over SYNCHRO. Although VISGAOST (Stevanovic et al., 2008) made some progress on arterial optimization, it is still a challenge to improve overall performance for a complex network without obvious traffic arteries.

For the two fixed signal timing plans produced by SYNCHRO, the results are $V_n^S = 4.99$ m/s and $T_w^S = 78.23$ s for “non-event”, and $V_n^S = 4.90$ m/s and $T_w^S = 80.98$ s for “midday”, respectively.

4.3.2. Local improvement

Although SYNCHRO produces coordinated signal plans that work well on microscopic traffic patterns, it is not able to utilize the nontrivial second-by-second variability in microscopic traffic flow information. As shown in Section 4.2.2, SchIC can be expected to outperform fixed coordination if the prediction horizon is not too short and the traffic demand is not too high. Thus, a natural extension is to locally control a small set of intersections to enhance real-time adaptivity while not impeding the coordination, given that all other intersections are still controlled by the fixed plans obtained by SYNCHRO.

[2] The turning movement counts of all intersections in both periods can be obtained from authors.

Table 4
Percent improvement over the fixed plans by SYNCHRO and computational cost of SchIC$_{G2}$ on the {8,10, 12}-intersection cases.

	"Non-event"				"Midday"			
	V_n	T_w	T_{cpu}	N_{rs}	V_n	T_w	T_{cpu}	N_{rs}
6-Intersections	5.69%	14.31%	0.255	10,366	5.27%	13.89%	0.311	11,836
8-Intersections	6.41%	16.28%	0.254	13,884	6.03%	16.07%	0.315	15,737
10-Intersections	7.96%	19.70%	0.319	17,473	7.36%	18.83%	0.384	19,559
12-Intersections	8.65%	21.90%	0.359	20,078	8.46%	21.76%	0.417	22,456

Table 5
Percent improvement over the fixed plans by SYNCHRO and computational cost of SchIC$_{G2}$ on the 6-intersection case with different G_{min} constraints.

	"Non-event"				"Midday"			
	V_n	T_w	T_{cpu}	N_{rs}	V_n	T_w	T_{cpu}	N_{rs}
G_{min} = 15	4.98%	12.55%	0.201	7822	4.58%	12.22%	0.237	8744
G_{min} = 20	3.90%	9.82%	0.157	6188	3.37%	9.24%	0.196	6908
G_{min} = 25	2.95%	7.31%	0.140	5225	2.52%	6.92%	0.165	5858

To investigate this possibility, we selected a small set of intersections to be adaptively controlled. For each intersection identified to be under adaptive control, we assumed that advance detectors were placed on the upstream end points of entry approaches. As a default, settings of G_{min} = 10 s and G_{max} = 60 s were used, and all Y values were kept the same as in the original settings. To select the subset of intersections to be adaptively controlled, a small set of runs on different time-of-day scenarios were made with each of the 32 intersections being the sole adaptive intersection. Any intersection yielding better network performance than that of the fixed plans generated using SYNCHRO when operated in traffic-adaptive mode was collected as a selection candidate, and the top n candidates were then incorporated as adaptively controlled intersections for the experiment.

Table 4 gives the performance on SchIC$_{G2}$ on the {6, 8, 10, 12}-intersection cases (the six adaptively controlled intersections are marked with ○, and additional selected intersections are incrementally marked with □ in Fig. 12). The 6-intersection case contains two three-phase intersections and four two-phase intersections. The percent improvement over the fixed plans produced by SYNCHRO are reported for V_n and T_w over the entire network, i.e., $\left(1 - V_n^S/V_n\right)$ and $\left(1 - T_w/T_w^S\right)$ respectively. The computation cost is also listed. Compared to the fixed plans produced by SYNCHRO, the percent improvement can be over 8% on V_n and over 20% on T_w, whereas the computational cost remains low.

Urban intersections can require long minimum green times for pedestrian crossing. To consider this factor, we also evaluated the performance of SchIC$_{G2}$ on the 6-intersection case with G_{min} = {15, 20, 25} s, as shown in Table 5. From the table, it can be seen that solution quality decreases at a moderate pace, but remains good even as G_{min} is 25 s.

It should be noted that not all intersections in a road network can be improved by the isolated application of adaptive control strategies. For tightly-coupled intersections, explicit coordination becomes critical for improving the overall performance. Nevertheless, our experiments do demonstrate that SchIC can work rather well on a large subset of intersections in a practical road network. Thus traffic engineers might adopt an economical strategy of incrementally introducing adaptive control to only a subset of intersections, and later implementing coordination mechanisms with other intersections (Mirchandani and Head, 2001; Gartner et al., 2002), if necessary.

5. Discussion

5.1. Basic characteristics

SchIC achieves encouraging performance through both heuristic space reduction and efficient state elimination. As an initial attempt to actively utilize structural information in traffic flow, research in this direction might provide greater understanding of the "true" intersection control problem, as compared to more domain-independent, "black-box" optimization, where structural information in traffic flow is for the most part ignored.

5.1.1. State space reduction

To avoid unnecessary risk of degrading solution quality, SchIC does not rely on the simple space reduction settings used in other existing methods such as a coarser time resolution (Shelby, 2004; Porche and Lafortune, 1999), a short optimization horizon (Robertson and Bretherton, 1974; Henry et al., 1983; Sen and Head, 1997), or a smaller number of phase switches (Gartner et al., 2002).

Central to SchIC is a scheduling-based formulation of the intersection control problem. In this formulation, vehicles are preprocessed into clusters based on the non-uniformly distributed nature of traffic flow. The intuition is that the cumulative delay is only associated with those vehicles that are delayed. The space $\mathcal{D}$ contains all feasible permutations of clusters/jobs, and feasible schedules are constructed iteratively by adding jobs one by one, with each new successor state being defined at the actual departure time of each job. As explained earlier, each decision can be viewed as a "heuristic leap" over multiple time steps in the space Ω, based on use of several simple heuristics. The heuristic "Do not split a cluster" follows the use of some aggregation patterns previously developed for reactive control strategies (Dunne and Potts, 1964; Viti and van Zuylen, 2010; Lämmer and Helbing, 2008), but the decision for servicing each cluster is instead made by a look-ahead optimization process that is able to avoid myopic mistakes. The heuristic "Do not explore in an idle time period" eliminates search through equivalently valued states in the case of forced idle time without loss of optimality. A region of these states provides no information for local search heuristics. Finally, the heuristic "Do not intentionally defer a scheduled cluster", eliminates consideration of dominated states in accordance with Theorem 1. Branch-and-bound methods (Kim et al., 2005; Porche and Lafortune, 1999), in contrast, might not detect such circumstances until a serious delay has been accumulated later on in the search tree.

Through simple heuristics tailored to flow information, the scheduling space $\mathcal{D}$ retains a large number of promising solutions with some are likely to be near optimal. Compared to a heuristic search technique, e.g., CRONOS, there is a stronger guarantee that the scheduling space $\mathcal{D}$ contains an (near) optimal solution in the underlying state space Ω.

5.1.2. State elimination

Exhaustive and branch-and-bound methods (Shelby, 2004; Kim et al., 2005; Porche and Lafortune, 1999) are not tractable in real time for realistic settings, although they can ensure optimality. Heuristic search techniques (Boillot et al., 2006) give no guarantee of achieving high-quality solutions, although they can attain high efficiency.

In existing state elimination techniques (or state equivalence relations (Shelby, 2004)), the state with the minimal cumulative delay d among a group of "equivalent" states is kept. For grouping states, both COP and PRODYN require the same time step and switched to the same phase index, i.e., (s, t). PRODYN also requires the same queue lengths on all lanes. However, PRODYN does not consider any arriving vehicles and the condition for equivalence of two states is very restrictive (unless a further approximation is introduced). COP also requires the same number of phase switches, which is only for the convenience of realizing its phase-wise stages. COP does not take queuing and arriving vehicles into account.

For efficient elimination of states at early stages of the search, states in SchIC are grouped together if they have the same (X, s). In the "full" *StateManager* mode, states in a group are maintained as a non-dominated set based on two state values, i.e., (t, d), to ensure optimality in the scheduling search space $\mathcal{D}$. In the "greedy" *StateManager* mode, a state group (X, s) degenerates into an approximate state equivalence relation.

For state elimination, a fundamental change in SchIC is the shift in focus from the finish time t to the schedule status X. As a pattern that emerges from the scheduling model, the complement of X indicates which vehicle clusters have not left the intersection on each route, which provides more accurate structural information on traffic flow. By using X, all states in a group are fairly compared since they have no difference in remaining vehicles.

5.1.3. Complexity and performance

The worst-case time complexity for SchIC has some interesting properties. As shown in Proposition 9, the "greedy" mode in the worst case will make $|I|^2 \cdot \prod_{i=1}^{|I|}\left(|H_P^{(i)}| + 1\right)$ state updates, and the "full" mode will in the worst case make $|H_O|$ times more updates than the "greedy" mode. The main calculation step of any given state update can be accomplished in constant time. As a bound for $|C^{(i)}|,\ |H_P^{(i)}|$ can be tighten by $H_P^{(i)}/(thc + \Delta)$ (based on Proposition 1), which is bounded if thc is large enough, and in this case the time resolution Δ can be arbitrarily fine. It is reasonable to set thc around or slightly above the saturation headway (shw), which in practice is several seconds. For the traffic control problem, an implicit feature is that the number of phases $|I|$ is bounded. Thus, SchIC has a polynomial time complexity in $|H_P|$, since the upper bound of $|H_O|$ is polynomial in $|H_P|$ (Proposition 7). Since the time complexity is mostly related to $|H_P|$, $|H_O|$ can be set to a large size so that a large number of schedules (or full-clearance states) are considered. This enables better advantage to be taken of the benefit of full-clearance optimization in obtaining better quality solutions. For saturated traffic conditions, H_O might be much larger than H_P to allow full clearance of vehicles. Note that the prediction horizon H_P might be extended in the case of explicit coordination between neighbor intersections (Mirchandani and Head, 2001; Gartner et al., 2002).

The average performance of SchIC in our experimental analysis is much better than the theoretic worse case. For example, as shown in Table 1, for an isolated intersection with $H_P/\Delta = 140$ and a demand of 1200 vph, the average number of state updates per roll step ($N_{ss/rs}$) of SchIC$_{G2}$ is only 43.3, which is close to the times required by some heuristic search techniques, e.g., CRONOS. Compared to COP2, SchIC$_{G2}$ obtained higher quality solutions and required $8.37\text{E}+05/4.33\text{E}+01 \approx 1.93\text{E}4$ less time on $N_{ss/rs}$. The average CPU time per roll step is $0.009/111 \approx 8.1\text{E}-5$ s. The average size of state groups can be as low as 1.31 (by comparing SchIC$_{F2}$ and SchIC$_{G2}$), which might be due to a high correlation between earlier finish times and lower cumulative delays in the scheduling search space $\mathcal{D}$.

5.2. Possible extensions

To keep the description and analysis of SchIC as simple as possible, some assumptions and simplifications have been made. For traffic control practice, some auxiliary extensions might be introduced to increase the overall scope of the real-world applicability.

5.2.1. Complex intersections

First, the number of phases $|I|$ in a cycle has been assumed to be very small. In our tests, almost all intersections have only two or three phases. However, some complex intersections can have substantially more. But the number of clusters $|C^{(i)}|$ on each route i can be further reduced by aggregating the anticipated queue based on a later start time, i.e., the minimal switching time from the initial phase to the phase i. On the route i, all clusters can be aggregated into an anticipated queue, i.e., $|C^{(i)}| = 1$, if the later start time is larger than $H_P^{(i)}$. This condition is easily satisfied for later phases since Y and G_{min} values are counted in Eq. 1. Thus, the computational cost will not increase much even if $|I|$ is large.

A complex design with a large number of phases might also be decomposed into a set of simple phase designs with less numbers of phases that are adaptively used in real time. The basic foundation is that the movements in any two entry approaches can be serviced in one or two phases with different combinations of movements. Since most real-world intersections have no more than four entry approaches, each simple phase design normally contains no more than four phases. Then in some roll steps, an auxiliary procedure can be included to select one of the simple phase designs for the intersection, based on analysis of major movements in recent traffic flow data. Overall performance might benefit more from a much shorter lost time in each cycle, than from the use of a complex phase design.

5.2.2. Splitting clusters

In making Assumption 1 that each cluster is non-divisible, the intuition is to avoid incurring the rather long setup time for clearing a residual queue. This intuition might not be valid for an extreme case where a cluster has a rather long duration and low flow rate as compared to a competing cluster that appears within another route (e.g., if one route has many fewer lanes than another route). Such a case might be easily addressed by applying a cap on the extension interval, so that the decision can be re-evaluated in the next roll step.

To avoid splitting clusters, the maximum green times are not included in the optimization procedure of SchIC. For a schedule, only if the departure time of the first job exceeds the maximum green time of the current phase, the repair rule will lead to a disturbance. The disturbance should not significantly reduce performance if the current extension services for vehicles in a sufficiently high flow rate. For the extension decision in Section 3.4, the idle period before the first job is guaranteed to be no longer than the minimum switchback time.

Moreover, it is possible to satisfy all constraints in the output schedule by only cutting some jobs. A greedy strategy is to check the schedule from the start, if any phase exceeds the maximum limit, then a cut is performed at the boundary, and all jobs after the cut point are rescheduled by SchIC. This strategy only needs to call SchIC a few times. Dynamic job cutting might be also integrated into SchIC, although the realization is more complicated.

5.2.3. Enhanced traffic models

For model-based optimization methods, a basic assumption is that the control model is sufficiently accurate for obtaining near optimal solutions of the real problem. The control model in SchIC is quite simple. The use of constant free-flow speed (Sharma et al., 2007; Mirchandani and Head, 2001) can capture the rationality of drivers. In principle, the speed variation details can be neglected once vehicles have joined an anticipated queue. The queue is cleared at the saturation flow rate after the start-up lost time (Sharma et al., 2007; Shelby, 2004; Sen and Head, 1997). If a delayed cluster has a flow rate lower than saturation, the cumulative delay is estimated to be higher than the actual value. However, our experimental results show that SchIC performs well with respect to solution quality. This might be due to leverage provided by the rolling horizon scheme, since only the first cluster in the solution (normally a queue with the saturation flow rate) is considered in the extension decision. This effect might also imply that those late-arriving clusters might be further aggregated since they are less important than the early-arriving ones. Nevertheless, more advanced traffic models can be easily embedded for better evaluation of states, if necessarily.

A related issue concerns the accuracy of input data, including measured flow data and model parameters. Perfect flow data can be obtained if the predicted arrival time and the requesting phase of each vehicle are known, as assumed in some methods (Sen and Head, 1997; Porche and Lafortune, 1999; Shelby, 2004). In practice, flow data are often obtained via surveillance cameras (Boillot et al., 2006) or sensors (Sharma et al., 2007; Mirchandani and Head, 2001). In this paper, only two detectors are used on both sides of each entry approach, thus some inaccuracy might exist in measured flow data. For example, vehicles might move at different speeds before they join into the anticipated queue, and right and left turns can only be handled through using estimated turning movement proportions. To counteract the risk of using static model parameters, a practical alternative is to use a moving-average estimation of each model parameter to handle the variability in different time periods and intersections (e.g., the saturation headway ranges from 1.8 to 2.4 s in the 2000 Highway Capacity Manual, whereas 2.5 s of slow-moving queues are used in Cai et al. (2009), Shelby (2004) and this paper). Nevertheless, as shown in Fig. 7, SchIC has good robustness under uncertainty, since the rolling horizon scheme ensures that errors are not accumulated over time. The detection at the stop line can be used for calibrating the clearance time of each cluster. In the future, accurate flow data might be obtained by advanced techniques, e.g., vehicle-to-infrastructure (V2I) communication, and thus real-time corrections might be made.

5.2.4. Explicit, network-level coordination

In isolated application, the capability of a model-based control strategy is limited since the local prediction horizon is bounded by the lengths of entry approaches. However, when employed as a core control strategy for individual intersections in a larger network, the myopic local view of each intersection can be strengthened significantly by a network-wide management system (Smith et al., 2007; Mirchandani and Head, 2001; Gartner et al., 2002). The basic interfaces for such network-level coordination are embedded in the inputs of the basic control framework defined in Fig. 1; i.e., non-local impacts from neighboring interesections might be incorporated by forecasting beyond the local prediction horizon of the route flow information and/or by projecting additional operating constraints (Mirchandani and Head, 2001; Gartner et al., 2002). For example, a moving average of previous arrivals along each approach is used in OPAC. In RHODES, the REALBAND decisions obtained in a sub-network are projected as the constraints to its intersection control strategy, i.e., COP. Traditionally offset length calculation has been projected as an additional operating constraint.

It is also possible to use explicit peer-to-peer exchange of schedule information to achieve coordination among neighboring intersections and amplify the performance of SchIC as a core intersection control strategy in complex road networks. In such a network, the schedule of an upstream intersection can be used to produce predicted output flows beyond the local prediction horizons of downstream neighbors. Due to the chain propagation, the look-ahead horizon extension of each intersection is capable of including non-local impacts from both direct and indirect upstream intersections. In comparison to other model-based intersection control methods, SchIC has a nontrivial advantage since the horizon extension does not significantly increase the computational complexity. The coordination can be perfect if all intersections essentially follow their schedules, and minor changes in schedules might still be absorbed by exploiting local temporal flexibility. Additional coordination mechanisms can be added to address dynamics, such as preventing instability due to spill back or reducing mis-coordination due to critical schedule changes of neighbors in different roll steps. In all cases, these mechanisms might exploit details in current (extended) intersection schedules (i.e., before they have been formally adopted). For example, a simple yet nontrivial mechanism is to force that each schedule satisfies the maximum green constraints. It is also possible to identify a spill back before it really happens, based on the detail flow information contained in each schedule. The efficiency of SchIC also allows itself to be iteratively invoked by more complex negotiation mechanisms to reach the global optimum in a sub-network.

6. Conclusions

In this paper, we have described a real-time intersection control strategy, called SchIC, for an adaptive traffic signal control. Based on an aggregate representation on predicted traffic flow data, the intersection control problem is formulated as a scheduling model, in which each schedule is a phase switching sequence that is generated utilizing structural information in non-uniformly distributed traffic flows. The scheduling model is then solved by a forward recursive algorithm with an efficient elimination criterion, which not only reduces the state space significantly in practice but also retains promising solutions. SchIC can efficiently find near optimal solutions in an extended optimization horizon with a fine time resolution.

We studied the performance of SchIC on two ideal scenarios and one real-world urban traffic network. For an isolated intersection with a long prediction horizon, SchIC was shown to obtain solutions with higher quality than both a vehicle-actuated logic (VA) and COP at lower computational expense, especially in the saturation region with high demands. For an arterial network, SchIC demonstrated the ability to achieve better implicit coordination between intersections, as compared to VA, COP, and a fixed timing plan with explicit coordination, if the prediction horizon is not too short and the demand is not too high. Given such a traffic control strategy with strong ability to achieve implicit coordination, a nontrivial implication is that a large traffic network might be easily decomposed into sub-networks. We also demonstrated that the performance of SchIC can be more robust than fixed coordination with respect to handling local variation in the traffic demand. A straightforward application was then demonstrated on a real-world urban traffic network. Based on the coordinated timing plans obtained by SYNCHRO for time-of-day periods with macroscopic traffic patterns, SchIC was applied to a subset of intersections to exploit microscopic variability in non-uniformly distributed traffic flow and was shown to significantly enhance the overall performance of the offline optimized network. Possible extensions for enhancing the overall scope of the real-world applicability of SchIC were also discussed.

Acknowledgements

This research was supported in part by the Traffic21 Initiative at Carnegie Mellon University, with support from the Hillman Foundation and the Heinz Endowments, and the CMU Robotics Institute. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

Bell, M.G., 1992. Future directions in traffic signal control. Transportation Research 26 (4), 303–313.

Boillot, F., Midenet, S., Pierrelee, J., 2006. The real-time urban traffic control system CRONOS: algorithm and experiments. Transportation Research Part C: Emerging Technologies 14 (1), 18–38.

Brockfeld, E., Kühne, R.D., Wagner, P., 2004. Calibration and validation of microscopic traffic flow models. Transportation Research Record 1876, 62–70.

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

Chase, C., Serrano, J., Ramadge, P.J., 1993. Periodicity and chaos from switched flow systems. IEEE Transactions on Automatic Control 38 (1), 70–83.

Cremer, M., Keller, H., 1987. A new class of dynamic methods for the identification of origin–destination flows. Transportation Research Part B: Methodological 21 (2), 117–132.

Du, J., Leung, J.Y.-T., 1990. Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 15 (3), 483–495.

Dunne, M.C., Potts, R.B., 1964. Algorithm for traffic control. Operations Research 12 (6), 870–881.

Gartner, N., Pooran, F., Andrews, C., 2002. Optimized policies for adaptive control strategy in real-time traffic adaptive control systemslu. Transportation Research Record 1811, 148–156.

Henry, J., Farges, J., Tufal, J., 1983. The PRODYN real time traffic algorithm. In: IFAC/IFIC/IFORS Conference on Control in Transportation System. pp. 305–310.

Husch, D., Albeck, J., 2003. Synchro 6 User Guide. Trafficware Ltd., Albany, CA.

Kamarajugadda, A., Park, B., 2003. Stochastic Traffic Signal Timing Optimization. Tech. Rep. UVACTS-15-0-44. Department of Civil Engineering, University of Virginia, Charlottesville, VA.

Kim, C.O., Park, Y., Baek, J.-G., 2005. Optimal signal control using adaptive dynamic programming. In: International Conference on Computational Science & Its Applications. Singapore, pp. 148–160.

Koulamas, C., 2010. The single-machine total tardiness scheduling problem: review and extensions. European Journal of Operational Research 202 (1), 1–7.

Krajzewicz, D., Hertkorn, G., Rssel, C., Wagner, P., 2002. SUMO (Simulation of Urban MObility): an open-source traffic simulation. In: Middle East Symposium on Simulation and Modelling. SCS European Publishing House, Sharjah, United Arab Emirates, pp. 183–187, <http://sumo.sourceforge.net>.

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

Luyanda, F., Gettman, D., Head, L., Shelby, S., Bullock, D., Mirchandani, P., 2003. ACS-Lite algorithmic architecture: applying adaptive control system technology to closed-loop traffic signal control systems. Transportation Research Record 1856, 175–184.

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

Newell, G.F., 1998. The rolling horizon scheme of traffic signal control. Transportation Research Part A: Policy and Practice 32 (1), 39–44.

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

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

Porche, I., Lafortune, S., 1999. Adaptive look-ahead optimization of traffic signals. ITS Journal 4 (3-4), 209–254.

Robertson, D.I., 1969. TRANSYT: A Traffic Network Study Tool. Tech. Rep. TRRL-LR-253. Transport and Road Research Laboratory, Crowthorne, UK.

Robertson, D.I., Bretherton, R.D., 1974. Optimum control of an intersection for any known sequence of vehicle arrivals. In: IFAC/IFIP/IFORS Symposium on Traffic Control and Transportation Systems. Monte Carlo, Monaco, pp. 3–17.

Robertson, D.I., Bretherton, R.D., 1991. Optimizing networks of traffic signals in real time – the SCOOT method. IEEE Transactions on Vehicular Technology 40 (1), 11–15.

Schrank, D., Lomax, T., Eisele, B., 2011. Annual Urban Mobility Report. Tech. Rep.. Texas Transportation Institute, Texas A&M University System, TX.

Sen, S., Head, K., 1997. Controlled optimization of phases at an intersection. Transportation Science 31 (1), 5–17.

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

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

Sims, A., Dobinson, K., 1980. The Sydney coordinated adaptive traffic (SCAT) system: philosophy and benefits. IEEE Transactions on Vehicular Technology 29 (2), 130–137.

Smith, S.F., Gallagher, A.T., Zimmerman, T.L., Barbulescu, L., Rubinstein, Z.B., 2007. Distributed management of flexible times schedules. In: International Conference on Autonomous Agents and Multiagent Systems. Honolulu, HI, pp. 484–491.

Stevanovic, J., Stevanovic, A., Martina, P.T., Bauer, T., 2008. Stochastic optimization of traffic control and transit priority settings in VISSIM. Transportation Research Part C 16 (3), 332–349.

Viti, F., van Zuylen, H.J., 2010. A probabilistic model for traffic at actuated control signals. Transportation Research Part C: Emerging Technologies 18 (3), 299–310.

Wu, X., Liu, H.X., Gettman, D., 2010. Identification of oversaturated intersections using high-resolution traffic signal data. Transportation Research Part C: Emerging Technologies 18 (4), 626–638.