Coordinated intersection traffic management

Pavankumar Tallapragada * **Jorge Cortés** *

* *Department of Mechanical and Aerospace Engineering, University of California, San Diego (e-mail:* {ptallapragada,cortes}@ucsd.edu*)*

Abstract: This paper considers the problem of coordinating the passage of vehicles through a traffic intersection with the aim of minimizing total travel time and energy consumption. The intersection manager communicates with vehicles heading towards the intersection, groups them into clusters (termed bubbles) as they appear, and determines an optimal order of passage and average velocity profiles. Vehicles in a bubble receive the corresponding profile and implement local control to avoid collision with other bubbles in the same road and within the bubble itself, and reach the intersection at the prescribed time and with the bubble occupying the intersection for no more than a prescribed duration.

Keywords: Intelligent transportation systems, hierarchical and distributed control, optimized operation and scheduling, state-based intersection management, networked vehicles

1. INTRODUCTION

Emerging technologies in intelligent transportation systems such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication have the potential to hugely impact safety, traveling ease, travel time, and energy consumption, eliminating road accidents and traffic collisions. A particularly useful application of these technologies is in the coordination of traffic at and near intersections. In contrast to traditional intersection management, networked vehicle technologies allow us to coordinate the traffic not just *within the intersection*, but also by controlling the vehicles' behavior much before they arrive at the intersection. Such a paradigm offers the possibility of significantly reduced stop times and increased fuel efficiency and is the subject of this paper.

Literature review

Much of the literature in the area of coordination-based intersection management focuses on collision avoidance of vehicles *within the intersection*. Supervisory intersection management (intervention only when required to maintain safety by avoiding collisions) is explored using discrete event abstractions in (Dallal et al., 2013) and reachable set computations in (Colombo and Del Vecchio, 2015; Hafner et al., 2013). The works (Dresner and Stone, 2008; Fajardo et al., 2011) and references therein describe a multiagent simulation approach in which, upon a reservation request from a vehicle, an intersection manager accepts or rejects the reservation based on a simulation. Each vehicle attempts to conform to its assigned reservation and if this is prediced not to be possible at any time, the reservation is canceled. (Kowshik et al., 2011) also uses a reservation-based system to schedule intersection crossing times. In addition, the paper also provides provably safe maneuvers for vehicle following in a lane as well as for crossing the intersection. Hult et al. (2015); Campos et al. (2014) use model predictive control based method to coordinate the intersection crossing by vehicles and obtain suboptimal solutions to a linear quadratic optimal control problem. In (Qian et al., 2014) a heuristic policy assigns priorities to the vehicles, while each vehicle applies a priority-preserving control and legacy vehicles platoon behind a computer-controlled car.

We note that the ability to efficiently coordinate diminishes as the vehicles get closer to the intersection. This is why here we take an expanded view of intersection management that looks at the coordinated control of the vehicles much before they arrive at the intersection. The above methods are not suited for this setup or would prove to be too computationally costly. An example of the expanded view of intersection management is (Miculescu and Karaman, 2014), in which a polling-systems approach is adopted to assign schedules, and then optimal trajectories for all vehicles are computed sequentially in order. Such optimal trajectory computations are costly and depend on other vehicles' detailed plans, and hence the system is not robust. Closer to this paper, the works (Jin et al., 2012, 2013) describe a hierarchical setup, with a central coordinator verifying and assigning reservations, and with vehicles planning their trajectories locally to platoon and to meet the assigned schedule. The proposed solution is based on multiagent simulations and a reservation-based scheduling (with the evolution of the vehicles possibly forcing revisions to the schedule), both important differences with respect to our approach. (Li et al., 2014) is a recent survey of traffic control with vehicular networks and provides other related references.

Statement of contributions

We propose a provably safe hierarchical intersection management system aimed at optimizing a combination of cumulative travel time and fuel usage. The proposed system is composed of three main aspects: (i) clustering to identify vehicles that must platoon before arriving at the intersection. We refer to such clusters of vehicles as *bubbles*; (ii) a branch-and-bound based scheduling algorithm that identifies the optimal schedule for a simplified cost function; (iii) a distributed control algorithm for the vehicles that ensures overall safety and guarantees that the actual intersection crossing schedule does not violate the prescribed schedule. Advantages of our proposed system include provably safe algorithms that do not require extensive simulations; dynamic clustering to account for

the arrival of new vehicles in the problem domain and reduce the computational load on the branch-and-bound algorithm, a feature that also makes the algorithm applicable to a varied range of traffic conditions; and a distributed algorithm for local vehicular control which guarantees the desired aggregated behavior of each bubble. Proofs are omitted for reasons of space and will appear elsewhere.

2. PRELIMINARIES

We present here some basic notation and concepts on graph theory used throughout the paper.

Notation

We let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}_0$ denote the set of real, nonnegative real, integer, positive integer, and nonnegative integer numbers, respectively. For a non-empty ordered list $\mathcal{S} = \{i_1, \ldots, i_s\}$, we let $|\mathcal{S}|$ denote the cardinality of $\mathcal{S}$. Further, $\mathcal{S}(i)$ denotes the i^{th} element of $\mathcal{S}$. Thus, $\mathcal{S}(|\mathcal{S}|)$ denotes the last element of $\mathcal{S}$. For convenience, we also use the notation $j \in \mathcal{S}$ ($j \notin \mathcal{S}$) to denote that j is (is not) an element of the ordered list $\mathcal{S}$. For two ordered lists $\mathcal{S}_1$ and $\mathcal{S}_2$, we let $\mathcal{S}_1 \setminus \mathcal{S}_2$ denote the ordered list of elements that belong to $\mathcal{S}_1$ but not to $\mathcal{S}_2$, while preserving the same order as in $\mathcal{S}_1$. We let the notation $[u]_{u_m}^{u_M}$ denote the number u lower and upper saturated by u_m and u_M ($u_m \leq u_M$), respectively, i.e.,

$$[u]_{u_m}^{u_M} \triangleq \min\{u_M, \max\{u_m, u\}\}$$

Graph theory

We review basic notions following the exposition in (Bullo et al., 2009). A digraph of order n is a pair $G = (V, E)$, where V is a set with n elements called nodes and E is a set of ordered pair of nodes called edges. A directed path is an ordered sequence of nodes such that any ordered pair of nodes appearing consecutively is an edge. A cycle is a directed path that starts and ends at the same node and that contains no repeated node except for the initial and the final one. A digraph is acyclic if it has no cycles. A directed (or rooted) tree is an acyclic digraph with a node, called root, such that any other node of the digraph can be reached by one and only one directed path starting at the root. If (i, j) is an edge of a tree, i is the parent of j, and j is the child of i. Given a tree, a subtree rooted at i is the tree that has i as its root and is composed by all of its successors in the original tree.

3. PROBLEM STATEMENT

Consider an intersection and the incoming traffic along four branches as shown in Figure 1. For simplicity, we assume that (i) there is a single lane in each direction, (ii) all vehicles are identical with length L, (iii) vehicles do not turn at the intersection, (iv) there are no sources or sinks for vehicles along the branches - all new traffic appears at the beginning of the branches and must cross the intersection. It is possible to avoid assumption (iii) and allow turning. However, the differing travel speeds when turning and going straight affects the computation of the intersection occupancy time. In order to keep the problem setup and notation simpler, we make assumption (iii).

The dynamics of a vehicle with label j is given by

$$\dot{x}_j^v(t) = v_j^v(t), \tag{1a}$$
$$\dot{v}_j^v(t) = u_j^v(t), \tag{1b}$$

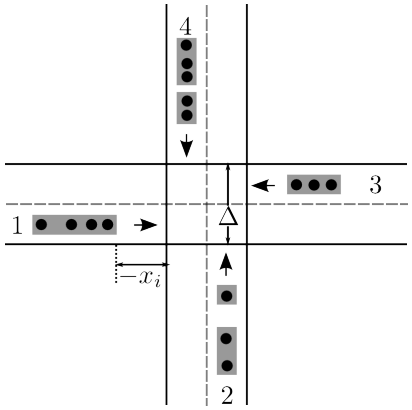

Fig. 1. Traffic near an intersection. Black dots represent individual vehicles, which are clustered and contained within *bubbles*, represented by grey boxes. Δ is the length of the intersection and the numbers $\{1, 2, 3, 4\}$ are labels for the incoming branches.

where x_j^v, $v_j^v \in \mathbb{R}$ are the position (negative of the distance from the front of the vehicle to the beginning of the intersection) and velocity of the vehicle, respectively and $u_j^v(t) \in [u_m, u_M]$, with $u_m \leq 0$ and $u_M \geq 0$, is the input acceleration. We use the superscript v for the state and control variables of individual vehicles. We assume that each branch has a maximum speed limit that the vehicles must respect. Purely for the sake of simpler notation, we assume that the speed limit on all branches is the same and equals v^M. Thus, for each vehicle j, $v_j^v(t)$ must be constrained to belong to the interval $[0, v^M]$ for all time t that the vehicle is in the system.

Each vehicle is equipped with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication capabilities. With the V2I communication, the vehicles inform a central *intersection manager* (IM) about their positions and velocities and receive from IM commands such as time to arrive at the intersection. We assume IM has the necessary communication and computing capabilities. We seek a design solution that minimizes a cost function $\mathcal{C}$, that models a combination of cumulative travel time and cumulative fuel cost, by scheduling the intersection crossing of the vehicles and controlling their approach to the intersection, all while avoiding collision. Solving this problem at the level of individual vehicles is computationally expensive and not scalable. Thus,we aim to synthesize a solution that makes this problem tractable to solve in real time and is applicable to a wide range of traffic scenarios.

4. OVERVIEW OF HIERARCHICAL SOLUTION

This section gives an outline of our hierarchical solution to the problem stated in Section 3. Our algorithmic solution combines optimized planning and scheduling of groups of vehicles with local distributed control to avoid collision and execute the plans, and has three distinct aspects,

(1) grouping the vehicles into clusters,
(2) scheduling the passage of the clusters through the intersection,
(3) local vehicular control to achieve and maintain cluster cohesion, to avoid collisions, and to ensure the clusters meet the prescribed schedule.

Each of these aspects is coupled with the other two. Moreover, an overarching theme is the dynamic nature of the problem due to the arrival and departure of vehicles.

Any complete or partial solution has to be computed as new vehicles come in (event based) or at regular time intervals (time based). For now though, we focus on the 'static' aspect of the solution and denote by t_0 an initial time at which the intersection manager 'samples' the state of traffic and solves the static scheduling problem. In what follows, we provide a general description of the main ingredients of each aspect.

Aspect 1 - clusters and bubbles. The primary motivation to cluster vehicles is to reduce the number of independent entities in the scheduling optimization problem. Clustering also has the advantage that, depending on the available computational resources, the maximum number of clusters could be fixed so that the more computationally expensive aspect of the solution (i.e., scheduling) remains scalable. At time t_0, the vehicles present in the four branches are grouped into N clusters, with N_i clusters on branch i. Given the position information of the vehicles at t_0, we use k-means clustering on each branch individually to identify the clusters (cf. Section 5).

The relative positions of the vehicles of a cluster may vary significantly over the course of their travel and the vehicles may not be in the form of a well-defined platoon at all times. Hence, we refer to a cluster of vehicles, by a generic term, as a *bubble* (shown as grey boxes in Figure 1). The defining characteristic of a bubble is that all the vehicles of a bubble cross the intersection uninterrupted. The state of the i^{th} bubble is given by the tuple

$$\xi_i = (x_i, v_i, m_i, \mathcal{I}_i) \in \mathbb{R}^3 \times \{1, 2, 3, 4\},$$

where x_i, v_i and m_i are, respectively, the position of the lead vehicle in the bubble, the velocity of the lead vehicle in the bubble, and the number of vehicles in the bubble. The quantity $\mathcal{I}_i$ denotes which of the four incoming branches the bubble is on. For each branch, we require the order of the bubbles to remain constant during the bubbles' travel (i.e., there is no passing allowed within each branch). To capture the order of the bubbles on a branch, we define the function $\mathcal{R}$,

$$\mathcal{R}(i,j) \triangleq \begin{cases} 1, & \text{if } \mathcal{I}_i = \mathcal{I}_j,\ x_j(t_0) < x_i(t_0), \\ & \nexists k \text{ s.t. } \mathcal{I}_k = \mathcal{I}_i,\ x_j(t_0) < x_k(t_0) < x_i(t_0), \\ 0, & \text{otherwise.} \end{cases}$$

In this definition, $\mathcal{R}(i,j) = 1$ if and only if bubble j and i are on the same branch and bubble j is the immediate follower of bubble i.

Aspect 2 - scheduling of bubbles. For the sake of simplicity, we restrict ourselves to only those schedules in which, at any given time, vehicles from a single incoming branch use the intersection. The job of the scheduler is to prescribe to each bubble an *approach time* τ_i - the time from t_0 that the i^{th} bubble takes to reach the beginning of the intersection, i.e., we require $x_i(t_0 + \tau_i) = 0$.

Aspect 3 - local vehicular control. The local vehicular control has various equally relevant goals. The first goal is to avoid collisions within each bubble and among different bubbles in the same branch. The second goal is for the local vehicular control to ensure that the bubble approaches the intersection in the prescribed time τ_i and that the *occupancy time* of the bubble, τ_i^{occ} (the time the intersection is occupied by bubble i), is no more than $\bar{\tau}_i^{\text{occ}}$, which in turn depends on m_i, the number of the vehicles in bubble i, and other system parameters and initial conditions. We assume that the control law at the vehicle level ensures that a vehicle does not change bubbles during the course of its travel time. Thus, as far as the scheduling aspect is concerned, m_i may be assumed constant in time.

Constraints. The preservation of the order of intersection crossing by the bubbles on the same branch takes the form,

$$\tau_j \geq \tau_i + \bar{\tau}_i^{\text{occ}}, \quad \text{if } \mathcal{R}(i,j) = 1, \tag{2a}$$

for $i, j \in \{1, \ldots, N\}$. Note that these constraints only ensure that the passage of bubbles on a branch through the intersection occurs in the same order as they have arrived, but they do not necessarily exclude collisions for the entire travel time. The intra-branch collisions are avoided at a local level and we accept the resulting sub-optimality. On the other hand, the no-collision constraint between bubbles on two different incoming branches takes the form,

$$\tau_i \geq \tau_j + \bar{\tau}_j^{\text{occ}} \text{ OR } \tau_j \geq \tau_i + \bar{\tau}_i^{\text{occ}}, \quad \text{if } \mathcal{I}_i \neq \mathcal{I}_j, \tag{2b}$$

for $i, j \in \{1, \ldots, N\}$. The constraints (2b) make the problem combinatorial in nature because of the need to determine whether i or j goes first. Since the order on each branch is to be preserved, the number of sub-problems is the number of permutations of the multiset $\{\mathcal{I}_i\}_{i=1}^N$, i.e.,

$$\frac{N!}{\prod_{i=1}^4 N_i!} = \frac{\left(\sum_{i=1}^4 N_i\right)!}{\prod_{i=1}^4 N_i!}, \tag{3}$$

where recall that N_i is the number of bubbles on branch i and N us the total number of bubbles.

5. DYNAMIC VEHICLE CLUSTERING

The primary motivation for clustering vehicles into bubbles is to reduce the computational burden on the scheduler. Consequently, we impose an upper bound, $\bar{\mathcal{N}}$, on the number of bubbles that the scheduler needs to consider at any given instance. Further, as new vehicles arrive, they need to be assigned to new bubbles. In order to balance both requirements, we divide each incoming branch into three zones: staging zone, mid zone and the exit zone, as shown in Figure 2. For each branch i, we let $\mathcal{Z}_i^s$, $\mathcal{Z}_i^m$ and $\mathcal{Z}_i^e$ be the set of positions on the branch i corresponding to the staging, mid and exit zones, respectively.

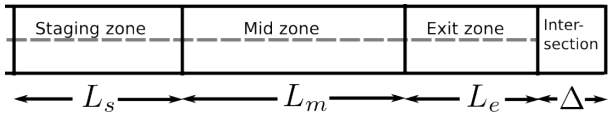

Fig. 2. Division of an incoming branch into zones.

The clustering algorithm is executed every $T_{cs} < \frac{L_s}{v^M}$ units of time, where L_s is the length of the staging zone and v^M is the max speed limit. At each clustering instance sT_{cs}, $s \in \mathbb{Z}$, the vehicles in the staging zone that do not already belong to a bubble are clustered. Thus, the choice $T_{cs} < \frac{L_s}{v^M}$ ensures that every vehicle belongs to a bubble before it leaves the staging zone and enters the mid zone. At a clustering instance sT_{cs}, let us suppose there are n_i^{ua} new vehicles to be clustered in the staging zone of branch i. Then, these vehicles are clustered based on their position using k-means algorithm with $k = \mathcal{M}_i = \min\{n_i^{ua}, \mathcal{N}_i\}$, where $\mathcal{N}_i$ is an upper bound on the number of new bubbles that may be created on branch i at the clustering instance.

At each instance sT_{cs}, the IM also schedules all the newly created bubbles as well as some or all of the previously created bubbles. If a bubble previously scheduled has already entered the exit zone, then its schedule is not modified any further. Similarly, scheduling is a costly

operation and we do not want the IM to schedule more than $\bar{\mathcal{N}}$ bubbles at any instance. Thus, if the number of newly created bubbles and the previously created bubbles yet to enter the exit zone exceeds $\bar{\mathcal{N}}$ then we pop out the required number of bubbles from the top of the list of bubbles previously scheduled and finalize their schedules. We present the precise algorithm to achieve this in Algorithm 1. The algorithm takes in the list of

Algorithm 1: `Clustering into bubbles` at sT_{cs}

Input: $\mathcal{L}_p$ {`Ordered list of bubbles scheduled at` $(s-1)T_{cs}$}
1: $\mathcal{L} \leftarrow \mathcal{L}_p \setminus \{j \in \mathcal{L} : \mathcal{I}_j = i \ \wedge \ x_j \notin \mathcal{Z}_i^s \cup \mathcal{Z}_i^m\}$
{`remove bubbles that have already entered exit zone`}
2: **for** $i = 1$ **to** 4 **do**
3: $\mathcal{N}_i$ {`max new bubbles on branch` i}
4: $\mathcal{M}_i \leftarrow \min\{n_i^{ua}, \mathcal{N}_i\}$ {`no. of new bubbles on branch` i}
5: Cluster new vehicles on branch i using k-means algorithm, with $k = \mathcal{M}_i$
6: **end for**
7: $\mathcal{M} \leftarrow \sum_i \mathcal{M}_i$
8: **if** $\mathcal{M} + |\mathcal{L}| > \bar{\mathcal{N}}$ **then**
9: Remove first $\mathcal{M} + |\mathcal{L}| - \bar{\mathcal{N}}$ bubbles from $\mathcal{L}$
10: **end if**
11: Append new bubbles to $\mathcal{L}$
12: $\tau^{\min} \leftarrow \max\left(\{\tau_p^{\min}\} \cup \{\tau_i + \bar{\tau}_i^{\text{occ}} : i \in \mathcal{L}_p \setminus \mathcal{L}\}\right)$
{`min. approach time for the bubbles` $\mathcal{L}$}

bubbles scheduled on the last iteration $\mathcal{L}_p$ and a minimum approach time $\tau_p^{\min}$ used when scheduling $\mathcal{L}_p$. The output of the algorithm is a list of bubbles $\mathcal{L}$ to be scheduled and the minimum approach time for them $\tau^{\min}$.

Remark 5.1. (On zone lengths). The lengths of the three zones illustrated in Figure 2 has a huge effect on the resulting traffic coordination. Systematic design of these zone lengths is out of the scope of this paper. However, we can identify some basic principles. Staging zone length L_s has a direct effect on the time step of the periodic execution of clustering and scheduling as well as on the number of vehicles per bubble. The mid zone length L_m has an effect on the likelihood of revising a bubble's schedule on the next iteration. The exit zone length L_e has an effect on the feasibility of the scheduling problem. For simplicity, we assume that L_e is large enough for a vehicle to come to a complete stop from a maximum speed of v^M in under a distance L_e. In any case, we envision these zone lengths to be of the order of several tens of meters. •

6. SCHEDULING OF BUBBLES

This section describes the scheduling algorithm employed by the intersection manager to decide the order of passage of the bubbles $\mathcal{L}$ through the intersection. This algorithm is also executed every T_{cs} units of time.

6.1 Cost function

We consider cost functions of the form

$$\mathcal{C} \triangleq \sum_{i=1}^{N} m_i(\tau_i + F_i(\bar{v}_i)) = \sum_{i=1}^{N} m_i\Big(\frac{d_i}{\bar{v}_i} + F_i(\bar{v}_i)\Big) \triangleq \sum_{i=1}^{N} \phi_i(\bar{v}_i), \tag{4}$$

where $\bar{v}_i$ is the average velocity of the lead vehicle in bubble i for $t \in [t_0, t_0 + \tau_i]$, i.e., $\bar{v}_i = \frac{d_i}{\tau_i}$, where $d_i \triangleq -x_i(t_0)$. The optimization variables are $\bar{v}_i$ (or equivalently τ_i) for each bubble i. Note that in the cost function $\mathcal{C}$, the functions F_i could, in general, depend on initial conditions as parameters - such as the distance to travel d_i. So, we see the cost function models a combination of cumulative travel time and total fuel usage.

As for the constraints, conditions on the travel times can be re-expressed as conditions on average velocities as

$$\tau_i \geq \tau_j + \bar{\tau}_j^{\text{occ}} \iff \frac{d_i}{\bar{v}_i} \geq \frac{d_j}{\bar{v}_j} + \bar{\tau}_j^{\text{occ}}$$
$$\iff \bar{v}_j \geq c_{ji}\bar{v}_i + b_{ji}\bar{v}_j\bar{v}_i, \quad c_{ji} = \frac{d_j}{d_i}, \ b_{ji} = \frac{\bar{\tau}_j^{\text{occ}}}{d_i}. \tag{5}$$

Thus, we re-express the no-collision constraints (2) as

$$\bar{v}_j \geq c_{ji}\bar{v}_i + b_{ji}\bar{v}_j\bar{v}_i \text{ OR } \bar{v}_i \geq c_{ij}\bar{v}_j + b_{ij}\bar{v}_i\bar{v}_j, \text{ if } \mathcal{I}_i \neq \mathcal{I}_j$$
$$\bar{v}_i \geq c_{ij}\bar{v}_j + b_{ij}\bar{v}_i\bar{v}_j, \quad \text{if } \mathcal{R}(i,j) = 1. \tag{6a}$$

In addition, we also need to ensure that τ_i for the bubbles scheduled at the instance sT_{cs} is no less than $\tau^{\min}$, defined in step 12 of Algorithm 1. Thus, for each bubble i, we have the constraint $\tau_i \geq \tau^{\min}$ or equivalently

$$\bar{v}_i \leq \frac{d_i}{\tau^{\min}}. \tag{6b}$$

Next, motivated by the fact that fuel efficiency is typically an increasing function of vehicle speed for speeds under the limits enforced at most intersections, we make the following assumption.

(A) For each i, $F_i : [\bar{v}_i^m, \bar{v}_i^M] \mapsto \mathbb{R}_{>0}$ is a monotonically decreasing function.

Note that the scheduling problem is combinatorial in nature due to the no-collision constraints. Thus, though the cost function $\mathcal{C}$ is somewhat simple and optimization variables are restricted to the average velocities $\bar{v}_i$, we believe it provides a good balance between usefulness and computational tractability. Further, the local vehicular control we present in Section 7 seeks an optimal control profile to achieve the prescribed average velocity for the bubble, which justifies the restriction to $\bar{v}_i$ as the optimization variables in the scheduling aspect. Thus our solution, although quite sub-optimal, is still principled.

We now focus on solving the problem of minimizing $\mathcal{C}$ in (4) under the constraints (6) and $\bar{v}_i \in [\bar{v}_i^m, \bar{v}_i^M]$. Note that the lower and upper limits on the average velocity, $\bar{v}_i^m$ and $\bar{v}_i^M$ respectively, depend on the initial conditions of the vehicles and desired speed limits and their computation is described in Section 7.1. Similarly, the upper bounds $\bar{\tau}_i^{\text{occ}}$ on the occupancy times may be computed as in Section 7.2. Now, we are ready to describe our solution to the scheduling problem. In the first part of the solution, we address the problem of determining the optimal schedule and optimal cost given a fixed order of bubble passage through the intersection. Then, we use a branch-and-bound algorithm to find the optimal order and schedule.

6.2 Optimal bubble average velocity given fixed order

Here we address the problem of determining, given a desired order of bubble passage through the intersection, the optimal average velocities of the bubbles and the associated optimal cost. For this purpose, define an *order* of the approach times of the bubbles as a permutation, P, of the integers from 1 to $|P| \leq N$. We use the notation $P(i)$ to denote the i^{th} element in the order, with the bubble $P(1)$ passing through the intersection first and so on. We use the notation $\sigma_P(i)$ to denote the position of bubble i

in the order P. Clearly, for a permutation to respect the intra-branch orders, $\sigma_P(i) < \sigma_P(j)$ if $\mathcal{R}(i,j) = 1$. Given an order P that respects the intra-branch orders, Algorithm 2 finds a solution to the optimization problem.

Algorithm 2: `Bubbles' velocity optimization`

Input: Order P
1: $C \leftarrow 0$
2: **for** $k = 1$ **to** $|P|$ **do**
3: $\quad i \leftarrow P(k)$ {`bubble` i `is in position` k `in` P}
4: $\quad$ **if** $k = 1$ **then**
5: $\quad\quad \bar{v}_i^P \leftarrow \bar{v}_i^M$
6: $\quad$ **else**
7: $\quad\quad j \leftarrow P(k-1)$ {`bubble` j `is in position` $k-1$ `in` P}
8: $\quad\quad \bar{v}_i^P \leftarrow \min\{\bar{v}_i^M, \frac{\bar{v}_j^P}{c_{ji} + b_{ji}\bar{v}_j^P}\}$
9: $\quad$ **end if** {$\bar{v}_i^P$ `is the optimizer for bubble` i}
10: $\quad C \leftarrow C + \phi_i(\bar{v}_i^P)$ {`update cost`}
11: **end for**

The following result shows that, for a fixed order P that respects the intra-branch order, the algorithm finds the average velocities that optimize the cost.

Lemma 6.1. (Algorithm 2 optimizes the schedule given an order P that respects the intra-branch orders). Consider the optimization of $\mathcal{C}$ (4) under assumption (A) and the constraints (6) and $\bar{v}_i \in [\bar{v}_i^m, \bar{v}_i^M]$, for $i \in P$. Suppose an order P respects the intra-branch orders. Then, $\bar{v}^P = (\bar{v}_1^P, \ldots, \bar{v}_N^P)$ and C given by Algorithm 2 are the optimizer and the optimum cost, respectively, for the order P.

6.3 Optimal ordering via branch-and-bound

We propose a branch-and-bound algorithm to solve the optimal scheduling problem. To describe the branching process, we introduce four queues (ordered lists), one for each branch. The queue for branch k, $Q_k = (i_{k,1}, \ldots, i_{k,N_k})$, is initialized to the list of all the bubbles on branch k in their order of arrival. Thus, $\mathcal{R}(i_{k,j}, i_{k,j+1}) = 1$ for all $j \in \{1, \ldots, N_k - 1\}$. We let $\mathcal{P}$ be any ordered list of up to length N with non-repeating numbers drawn from $\{1, \ldots, N\}$ and preserving the required individual branch orders. Thus, $\mathcal{P} = \emptyset$ (the empty list), denotes the root of the tree representing all feasible orders. $\mathcal{P} = (i_1, \ldots, i_k)$ denotes the subtree of all the feasible orders in which bubble i_1 crosses the intersection first, so on until bubble i_k is the k^{th} to cross and with the rest of the order undetermined.

Now, notice that step 8 in Algorithm 2 updates the higher limit on the feasible $\bar{v}_i$ given the order of all the bubbles preceding it. We could update/improve the higher limit even if we knew only part of the order preceding a bubble. We denote the higher limit on $\bar{v}_i$ given that a non-empty $\mathcal{P}$ precedes bubble i by $H_i^{\mathcal{P}}$ and it is given by Algorithm 3. Now, given $\mathcal{P}$ we can lower bound the optimal cost for any order in the subtree $\mathcal{P}$ in terms of $\bar{v}_i^{\mathcal{P}}$ and $H_i^{\mathcal{P}}$ as

$$C^{\mathcal{P}} \triangleq \sum_{i \in \mathcal{P}} \phi_i(\bar{v}_i^{\mathcal{P}}) + \sum_{i \notin \mathcal{P}} \phi_i(H_i^{\mathcal{P}}). \tag{7}$$

With this, we can now implement a branch-and-bound algorithm to find an optimal schedule for the bubbles.

The branch-and-bound algorithm starts by picking a candidate order, computing the cost for it, using Algorithm 2, and storing the two as the current best solution and cost. Then, starting at the root node of the tree of all feasible orders, the algorithm searches (e.g. depth-first or breadth-first) for an optimal solution. If at any time a leaf node, which corresponds to a complete order, is reached and its cost is better than the current best, then the current best solution and cost are updated. For any other node $\mathcal{P}$ in the tree, (7) provides a lower bound $C^{\mathcal{P}}$ on the cost of all the orders represented by the node $\mathcal{P}$. And if $C^{\mathcal{P}}$ is greater than the current best cost then the subtree $\mathcal{P}$ may be safely disregarded. In this way, the algorithm eventually finds an optimal solution.

Algorithm 3 : `Upper bound on optimal average velocity of a bubble, given list of bubbles preceding it`

1: $l \leftarrow \mathcal{P}(|\mathcal{P}|)$ {l `is last bubble in` $\mathcal{P}$}
2: Compute $\bar{v}_l^{\mathcal{P}}$ using Algorithm 2
3: **for** $k = 1$ **to** 4 **do**
4: $\quad \mathcal{Q}_k \leftarrow Q_k \setminus \mathcal{P}$ {`pop-out` $\mathcal{P}$ `from` Q_k}
5: $\quad$ **if** $\mathcal{Q}_k \neq \emptyset$ **then**
6: $\quad\quad i \leftarrow \mathcal{Q}_k(1)$ {i `is first of remaining bubbles in` $\mathcal{Q}_k$}
7: $\quad\quad H_i^{\mathcal{P}} \leftarrow \min\{\bar{v}_i^M, \frac{\bar{v}_l^{\mathcal{P}}}{c_{li} + b_{li}\bar{v}_l^{\mathcal{P}}}\}$
8: $\quad\quad$ **for** $s = 2$ **to** $|\mathcal{Q}_k|$ **do**
9: $\quad\quad\quad i \leftarrow \mathcal{Q}_k(s)$
10: $\quad\quad\quad j \leftarrow \mathcal{Q}_k(s-1)$
11: $\quad\quad\quad H_i^{\mathcal{P}} \leftarrow \min\{\bar{v}_i^M, \frac{H_j^{\mathcal{P}}}{c_{ji} + b_{ji}H_j^{\mathcal{P}}}\}$
12: $\quad\quad$ **end for**
13: $\quad$ **end if**
14: **end for**

7. LOCAL VEHICULAR CONTROL

The control at the local vehicular level broadly involves two tasks - the first is to compute the parameters $\bar{v}_i^m$, $\bar{v}_i^M$ and $\bar{\tau}_i^{\text{occ}}$ of bubble i for the scheduler; and the second is to control the vehicles so that all the vehicles of bubble i cross the intersection within the time interval $[\tau_i, \tau_i + \bar{\tau}_i^{\text{occ}}]$ that is prescribed by the scheduler. Successful execution of each of these tasks requires a better understanding of the coupled dynamics of the vehicles with desired safety constraints. To be precise, we introduce the following definitions of the maximum braking maneuver and a safe-following distance.

Definition 7.1. (Maximum braking maneuver (MBM)). The MBM, for a vehicle j, is a control action with $u_j^v = u_m$ until the vehicle comes to a stop and $u_j^v = 0$ thereafter. •

Definition 7.2. (Safe-following distance). Let $j-1$ and j be the indices of two vehicles on the same branch, with vehicle j immediately following $j-1$. We say a quantity $\mathcal{D}(v_{j-1}^v(t), v_j^v(t))$ is *a safe-following distance* at time t for the pair of vehicles $j-1$ and j if $x_{j-1}^v(t) - x_j^v(t) \geq \mathcal{D}(v_{j-1}^v(t), v_j^v(t))$ and if each of the two vehicles were to perform the MBM then the two vehicles would be safely separated ($x_{j-1}^v - L \geq x_j^v$) for all future. •

Note that according to this definition a safe-following distance is not uniquely defined, which in fact provides a certain leeway in designing the local vehicle control. Given this definition, we can now precisely quantify a safe-following distance as in the following lemma.

Lemma 7.3. (A safe-following distance). Let $j-1$ and j be labels of a pair of vehicles, with j following $j-1$. Then, a safe-following distance for j as a function of the velocities of the pair of vehicles is

$$\mathcal{D}(v_{j-1}^v(t), v_j^v(t)) = L + \max\left\{0, \frac{-1}{2u_m}\left((v_j^v(t))^2 - (v_{j-1}^v(t))^2\right)\right\}, \tag{8}$$

where recall that L is the vehicle length.

Remark 7.4. (Vehicle indices). In the remainder of this section, we index the vehicles in bubble i as $(i,1),\ldots,(i,m_i)$, where $(i,1)$ refers to the lead vehicle in bubble i and so on until (i,m_i), the last vehicle in the bubble. We also find it convenient for the label $(i,0)$ to represent the last vehicle $(i',m_{i'})$ of the bubble i' that precedes bubble i on the same branch or if there is no bubble preceding bubble i on the branch then we let $(i,0)$ be an imaginary vehicle located at ∞. We drop the index i whenever there is no ambiguity with regard to the bubble. •

7.1 Lower and upper limits on average velocity

Recall that $\bar{v}_i$ is the average velocity of the lead vehicle of bubble i from t_0 and until the lead vehicle reaches the beginning of the intersection at τ_i. Thus, it would seem that computing bounds on the achievable average velocity of the lead vehicle in the bubble is sufficient to determine $\bar{v}_i^M$ and $\bar{v}_i^m$. However, ignoring the initial conditions of the other vehicles in the bubble in the computation of $\bar{v}_i^M$ and $\bar{v}_i^m$ poses the risk of lengthening a guaranteed upper bound on the occupancy time, $\bar{\tau}_i^{\text{occ}}$. Thus, we propose the following alternative solution. In bubble i, for each vehicle (i,k), we let $\tau_{i,k}^m$ be the earliest time vehicle (i,k) can reach the intersection ignoring the other vehicles on the branch. The quantity $\tau_{i,k}^m - t_0$ is the time it takes x_i^v to reach 0 from $x_i^v(t_0)$ for the trajectory with maximum acceleration until $v_i^v = v^M$ and zero acceleration thereafter. Assuming a nominal speed ν^{nom} for vehicles when entering the intersection, we define $\mathcal{D}^{\text{nom}} \triangleq \mathcal{D}(\nu^{\text{nom}}, v^M)$, which has the connotation of a safe inter-vehicle distance given a vehicle is traveling at the maximum allowed speed v^M and the vehicle leading it traveling at a speed higher than ν^{nom}. Then, we also define $T^{\text{nom}} \triangleq \mathcal{D}^{\text{nom}}/\nu^{\text{nom}}$ as the *nominal inter-vehicle arrival time*. Then, we define *earliest time of arrival* of the bubble i, τ_i^m as

$$\tau_i^m \triangleq \max\{\tau_{i,k}^m - (k-1)T^{\text{nom}} : k \in \{1,\ldots,m_i\}\} \quad (9)$$

and let $\bar{v}_i^M = \frac{-x_i(t_0)}{\tau_i^m}$. Analogous computations with maximum deceleration yield the *latest time of arrival* of the bubble i, τ_i^M and $\bar{v}_i^m$. However, to guarantee the feasibility of the scheduling problem in a simple fashion, we assume that the exit zone length L_e is large enough for $\bar{v}_i^m$ to be zero (τ_i^M to be infinity).

7.2 Upper bound on guaranteed occupancy time

The idea for computing $\bar{\tau}_i^{\text{occ}}$ is similar to that used in the computation of $\bar{v}_i^M$. However, the local vehicular control may not be able to achieve inter-vehicular distances strictly upper bounded by $\mathcal{D}^{\text{nom}}$ when crossing the intersection. Instead we allow the inter-vehicle distances to be upper bounded by $\sigma_0\mathcal{D}^{\text{nom}}$ where $\sigma_0 > 1$ is a design parameter. Thus, we obtain

$$\bar{\tau}_i^{\text{occ}} = (m_i - 1)\sigma_0 T^{\text{nom}} + \frac{L+\Delta}{\nu^{\text{nom}}}. \quad (10)$$

7.3 Local vehicular control

For a bubble i, the scheduler prescribes a time τ_i at which the vehicles in bubble i may start to cross the intersection. The local vehicular control must ensure that the vehicles of bubble i start and finish crossing the intersection within the time interval $[\tau_i, \tau_i + \bar{\tau}_i^{\text{occ}}]$, respecting the safety constraints (8). In this subsection, we describe an algorithm to achieve this task. The algorithm has two main parts - an uncoupled controller that ensures the vehicle arrives at the intersection at a designated time ignoring other vehicles, which is applied when the precedent vehicle is sufficiently far in front; and a controller that ensures the vehicle follows the precedent safely otherwise; and a rule to switch between the two.

Uncoupled controller. Let us first define, for each vehicle $k \in \{1,\ldots,m_i\}$ in bubble i,

$$\tau_{i,k} \triangleq \tau_i + (k-1)T^{\text{nom}}. \quad (11)$$

Note that by design $\tau_i \in [\tau_i^m, \tau_i^M]$, which together with (9) implies that $\tau_{i,k} \in [\tau_{i,k}^m, \tau_{i,k}^M]$. Now, let

$$(t, x_{i,k}^v, v_{i,k}^v) \mapsto g_{uc}(\tau_{i,k}, t, x_{i,k}^v, v_{i,k}^v)$$

be a feedback controller that ensures that for the dynamics (1) $x_{i,k}^v(\tau_{i,k}) = 0$ starting from the current state $(x_{i,k}^v(t), v_{i,k}^v(t))$ at time t (assuming feasibility), respecting the control and velocity constraints but not necessarily the inter-vehicle safety constraints. Such a controller exists for each vehicle at least at $t = t_0$ due to the fact that $\tau_{i,k} \in [\tau_{i,k}^m, \tau_{i,k}^M]$. In this paper, we let g_{uc} be defined as the optimal feedback controller that generates velocity profiles as shown in Figure 3 obtained by optimizing $\int_{t_0}^{\tau} |u_j^v(s)|ds$, with $\tau = \tau_{i,k}$ and the optimization variables a_1, a_2, $v_j^v(\tau)$ and ν^l (or ν^u), where a_1 and a_2 are the areas of the indicated triangles. Among the constraints are that $v_j^v(\tau) \geq \nu^{\text{nom}}$ and that the total area under the curve must be equal to $-x(t_0)$. The feedback controller may be found by tabulating the optimal control solution.

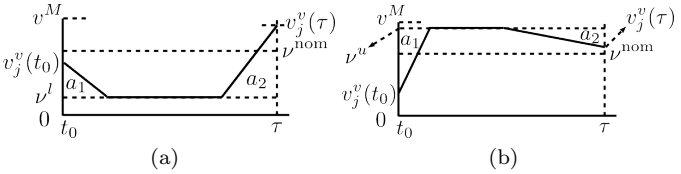

Fig. 3. Candidate velocity profiles to obtain g_{uc}.

Controller for safe following. As mentioned earlier, this controller is applied only when a vehicle is sufficiently close to the vehicle preceding it. Besides maintaining a safe following distance, the controller must also ensure that the resulting evolution of the vehicles in the bubble i is such that the occupancy time is no more than $\bar{\tau}_i^{\text{occ}}$. Here, we present a design to achieve these goals. For a pair of vehicles $j-1$ and j, with j following $j-1$, we let

$$\sigma_j(t) \triangleq \frac{x_{j-1}^v(t) - x_j^v(t)}{\mathcal{D}(v_{j-1}^v(t), v_j^v(t))}. \quad (12)$$

Notice from (8) that $\mathcal{D}$ is a continuous function of the vehicle velocities v_{j-1}^v and v_j^v and that if $v_{j-1}^v(t) > v_j^v(t)$ then σ_j only increases and safety is guaranteed. Thus, it is sufficient to design a controller that ensures safe following when $v_j^v(t) \geq v_{j-1}^v(t)$. For a vehicle $j \in \{1,\ldots,m_i\}$ of bubble i, consider the controller

$$g_{sf}(t, \sigma_j, v_{j-1}^v, v_j^v, u_{j-1}^v) \triangleq \min\{g_{uc}(\tau_{i,k}, t, x_{i,k}^v, v_{i,k}^v), \left[g_{us}(\sigma_j, v_{j-1}^v, v_j^v, u_{j-1}^v)\right]_{u_m}^{u_M}\}, \quad (13)$$

where g_{us} is the unsaturated controller

$$g_{us}(\sigma_j, v^v_{j-1}, v^v_j, u^v_{j-1}) \triangleq \left(\frac{v^v_{j-1}}{v^v_j}\left(1+\sigma_j\frac{u^v_{j-1}}{-u_m}\right)-1\right)\left(\frac{-u_m}{\sigma_j}\right).$$

Switching controller. Now, we specify the rule to switch between the controllers g_{uc} and g_{sf} or in other words the switching controller. We let the control for vehicle j be

$$u^v_j(t) = \begin{cases} g_{uc}, & \text{if } (v^v_{j-1}, v^v_j, \sigma_j) \notin \mathcal{C}_s \\ g_{sf}, & \text{if } (v^v_{j-1}, v^v_j, \sigma_j) \in \mathcal{C}_s \end{cases}, \qquad (14)$$

where $\mathcal{C}_s$ is the *coupling set* given by

$$\mathcal{C}_s \triangleq \{(v^v_{j-1}, v^v_j, \sigma_j) : v^v_j \geq v^v_{j-1} \ \wedge \sigma_j \in [1, \sigma_0]\} \qquad (15)$$

with $\sigma_0 > 1$ a design parameter. Then, we can show the following result, which says that the controller (14) ensures safety and satisfies the prescribed schedule.

Theorem 7.5. (Provably safe traffic coordination). Consider the vehicle dynamics (1). If a bubble precedes bubble i on its branch, then suppose the vehicle $(i, 0)$ (see Remark 7.4) reaches the intersection within its designated time and while crossing the intersection has a velocity of at least ν^{nom}. Then, for each vehicle (i, j), $j \in \{1, \ldots, m_i\}$ in the bubble i and for all $t \in [t_0, \tau_i + \bar{\tau}^{\text{occ}}_i]$ (i) the controller (14) is feasible and (ii) inter-vehicle safety is ensured, i.e., $\sigma_j(t) \geq 1$. In addition, the bubble crosses the intersection within the interval $[\tau_i, \tau_i + \bar{\tau}^{\text{occ}}_i]$. ■

8. CONCLUSIONS

We have studied the problem of coordinating traffic at an intersection in order to reduce travel time and improve vehicle energy efficiency while avoiding collisions. Our intersection management solution relies on communication among vehicles and the infrastructure, and combines hierarchical and distributed control to optimally schedule the passage of vehicle clusters or bubbles through the intersection. Our dynamic bubble-based approach has the advantage of reducing the complexity of the computationally intensive scheduling problem and making the solution applicable for different traffic conditions. Future work will explore the computational complexity of the proposed algorithm, incorporation of privacy preservation requirements, comparison with classical stop-and-go policies, and the extension to coordinated management for networks of intersections. Preliminary simulations (not included here due to lack of space) verify our proposed algorithm and in future further simulations will be performed to study the performance of the algorithm and to compare with traditional intersection management solutions.

ACKNOWLEDGMENTS

This research was partially supported by NSF Award CNS-1446891.

REFERENCES

Bullo, F., Cortés, J., and Martínez, S. (2009). *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press. Electronically available at http://coordinationbook.info.

Campos, G.R., Falcone, P., Wymeersch, H., Hult, R., and Sjöberg, J. (2014). Cooperative receding horizon conflict resolution at traffic intersections. In *IEEE Conf. on Decision and Control*, 2932–2937. Los Angeles, CA.

Colombo, A. and Del Vecchio, D. (2015). Least restrictive supervisors for intersection collision avoidance: A scheduling approach. *IEEE Transactions on Automatic Control*, 60(6), 1515–1527.

Dallal, E., Colombo, A., Del Vecchio, D., and Lafortune, S. (2013). Supervisory control for collision avoidance in vehicular networks using discrete event abstractions. In *American Control Conference*, 4380–4386. Washington, D.C.

Dresner, K. and Stone, P. (2008). A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31, 591–656.

Fajardo, D., Au, T., Waller, S.T., Stone, P., and Yang, D. (2011). Automated intersection control. *Transportation Research Record: Journal of the Transportation Research Board*, 2259, 223–232.

Hafner, M.R., Cunningham, D., Caminiti, L., and Del Vecchio, D. (2013). Cooperative collision avoidance at intersections: Algorithms and experiments. *IEEE Transactions on Intelligent Transportation Systems*, 14(3), 1162–1175.

Hult, R., Campos, G., Falcone, P., and Wymeersch, H. (2015). An approximate solution to the optimal coordination problem for autonomous vehicles at intersections. In *American Control Conference*, 763–768. Chicago, IL.

Jin, Q., Wu, G., Boriboonsomsin, K., and Barth, M. (2012). Advanced intersection management for connected vehicles using a multi-agent systems approach. In *IEEE Intelligent Vehicles Symposium*, 932–937. Alcalá de Henares, Spain.

Jin, Q., Wu, G., Boriboonsomsin, K., and Barth, M. (2013). Platoon-based multi-agent intersection management for connected vehicle. In *IEEE International Conference on Intelligent Transportation Systems*, 1462–1467. The Hague, Holland.

Kowshik, H., Caveney, D., and Kumar, P. (2011). Provable systemwide safety in intelligent intersections. *IEEE Transactions on Vehicular Technology*, 60(3), 804–818.

Li, L., Wen, D., and Yao, D. (2014). A survey of traffic control with vehicular communications. *IEEE Transactions on Intelligent Transportation Systems*, 15(1), 425–432.

Miculescu, D. and Karaman, S. (2014). Polling-systems-based control of high-performance provably-safe autonomous intersections. In *IEEE Conf. on Decision and Control*, 1417–1423. Los Angeles, CA.

Qian, X., Gregoire, J., Moutarde, F., and Fortelle, A.D.L. (2014). Priority-based coordination of autonomous and legacy vehicles at intersection. In *IEEE International Conference on Intelligent Transportation Systems*, 1166–1171. Qingdao, China.