Cloud-based Collision-Aware Energy-Minimization Vehicle Velocity Optimization

Chenxi Qiu
Department of Computer Science
Rowan University
Email: qiu@rowan.edu

Haiying Shen
Department of Computer Science
University of Virginia
Email: hs6ms@virginia.edu

***Abstract*—In recent years, many efforts have been devoted to reducing vehicles' energy consumption through optimizing their velocities. However, all previous methods neglect avoiding vehicle collision when calculating vehicles' optimal velocity profiles. In this paper, we formulate a new problem, called the *Collision-Aware vehicle Energy consumption Minimization (CAEM)* problem that calculates the optimal velocity profiles which avoid vehicle collision. CAEM is more difficult to solve than the traditional velocity optimization problem that is for a single vehicle, since CAEM needs to take into account the mobility of all the vehicles together to avoid collision. This problem is a convex problem that cannot be directly solved by existing methods. Further, it is impractical to get the mobility information of all vehicles at the beginning. Even if it is feasible, the computation efficiency is very low. We propose a novel method that can tackle these challenges. In order to keep each vehicle velocity as stable as possible to reduce the energy consumption, it builds a light schematic map to help identify the green light time interval of each traffic light in the source-destination route of a vehicle, during which the vehicle must drive through the traffic light. Rather than considering the mobility of all vehicles at a time, it calculates each vehicle's velocity profile in sequence based on the starting time to prevent the vehicle from colliding with all the previously scheduled vehicles. Finally, CAEM is transformed to non-convex problem and can be solved by existing optimization methods. Simulation and real-world testbed experimental results demonstrate the superior performance of our method over the previous methods.**

***Index Terms*—energy consumption, vehicle, velocity optimization**

I. Introduction

As the number of vehicles on the roads increases rapidly worldwide (e.g., more than 13 million vehicles were sold in China in 2014), decreasing vehicle energy consumption has been considered as an extremely important issue for the transportation system. The energy minimization for fuel vehicles saves the fuel energy and protects the environment. It is also important for electric vehicles, which are expected to replace the fuel vehicles in the future. Their batteries can only support a driving distance range from 80 to 100 miles [10], and hence need energy minimization to avoid frequent charging. Among various strategies to reduce the energy consumption of vehicles, optimizing vehicle velocity is one of the most effective methods [2], [10], [11], [13]–[15], [18]. This method outputs an optimal vehicle velocity profile that indicates the vehicle's velocity at each time point from its source location to its destination location to minimize its energy consumption.

Since more accelerations generate higher energy consumption [5], the objective of velocity optimization is essentially to reduce the number of accelerations. Some methods [2], [14], [18] rely on road side units (RSUs) to calculate a vehicle's velocity profile from its source to its destination. Other methods [10] let each vehicle communicate with traffic signals in order to approach traffic signals at green, whenever possible, and hence reduce the number of accelerations. Considering that the calculation of the optimal velocity profile that minimizes the energy consumption of a vehicle in its whole route is beyond the computing capability of either RSUs or traffic lights, a computing framework for transportation systems, called vehicular cloud, has been proposed [11], [13], [15]. In this framework, each vehicle uploads its information (starting time, travel time constraint, and route) to the cloud through base stations, and then the cloud derives the optimal velocity profile for each vehicle.

Although many efforts have been devoted to optimizing vehicle velocity to reduce energy consumption, most of these strategies consider each vehicle as an independent object and neglect the influence between consecutive vehicles in single lanes. Although a vehicle may pass its preceding vehicle through a neighboring lance, the influence between the consecutive vehicles in a single lane cannot be neglected, since most vehicles need to travel through single lanes in reality. For example, in the vehicle mobility trace from the Cologne urban area [1], we found that more than 80% vehicles must go through a single lane in their trips. Also, it is important to avoid changing lanes to pass the preceding vehicles since it complicates driving and increases the possibility of collision.

Therefore, when calculating vehicles' optimal velocity profiles to minimize individuals' energy consumption, it is important to additionally consider the influence between the consecutive vehicles in single lanes, so that vehicles will not collide if they follow their velocity profiles. Accordingly, in this paper, we formulate a new problem that has not been studied previously, namely the *Collision-Aware vehicle Energy consumption Minimization (CAEM)* problem. It aims to calculate the optimal velocities of all the vehicles in a vehicle transportation system that covers a region. One solution to this CAEM problem is that after the cloud receives the information from all vehicles, it calculates the optimal velocity profile of each vehicle that also avoids collisions between the vehicles

in single lanes. Vehicle drivers always hope to receive the velocity profiles quickly without much delay. However, this solution is not effective due to the following two challenges: 1) The constraints (i.e., including the red traffic light avoidance constraint and the vehicle collision awareness constraint) in this problem are non-convex [8], which means that this problem cannot be directly solved using the existing methods. 2) It is impractical for the cloud to receive the entire inputs at the beginning since there are always new vehicles joining the system (i.e., starting new travels or entering the region). Also, updating the velocity profiles of the existing vehicles that are affected by a newly joined vehicle generates a high overhead and a long delay.

As a solution, we propose a novel solution for CAEM, called *CAEM-S*. To tackle the first challenge, we can simplify the problem by exploiting the property of the vehicle energy consumption model [5], [6]. Specifically, we first theoretically derive that minimizing the energy consumption of a vehicle is essentially keeping the vehicle's velocity in a stable level, though previous works intuitively made this observation [2], [13]. Based on this principle, we can keep every vehicle's velocity constant in each route segment (i.e., a route without traffic lights, stop signs, or any other route connected to it) and reduces the velocity difference among different route segments by identifying the green traffic signal time interval (e.g., [8:00am-8:10am]) of each traffic light in the source-destination route of a vehicle, during which the vehicle should drive through the traffic light. Finally, the CAEM problem is simplified to a convex problem, which can be efficiently solved by the existing optimization methods [8].

To tackle the second and third challenges, we propose to calculate each vehicle's velocity profile in sequence in the order of starting time. That is, the cloud stores all the calculated velocity profiles in a table, called the *scheduled velocity table* (or simply *SV-Table*). For each newly joined vehicle, the cloud calculates its optimal velocity profile that avoids the collision with the previously joined vehicles based on the SV-table. In this way, it also tackles the third challenge since the cloud only needs to calculate one vehicle's velocity at a time.

Finally, we measure the performance of our method in both simulation and real-world testbed experiment. The experimental results demonstrate that our method outperforms two previous methods: dynamic programming (DP) [13] and predictive cruise control (PCC) [2], in terms of the number of velocity violations, energy consumption and computation delay. In summary, our contribution in this paper is threefold: 1) We formulate a new vehicle velocity optimization problem, called the CAEM problem, which avoids collision between the consecutive vehicles in single lanes.
2) To solve the CAEM problem, we propose CAEM-S that calculates vehicles' optimal velocity profiles in sequence in order to avoid colliding with already scheduled vehicles. To improve the time efficiency of CAEM-S, we first theoretically derive the constant velocity principle for energy minimization, and then propose a novel LSMap-based method that helps find constant velocity profiles. Then, the CAEM problem is simplified to a non-convex problem, which can be solved by the existing methods with low time complexity.
3) We measure the performance of our method using both simulation and real-world testbed experiments. The experimental results demonstrate the superior performance of our method over the previous methods.

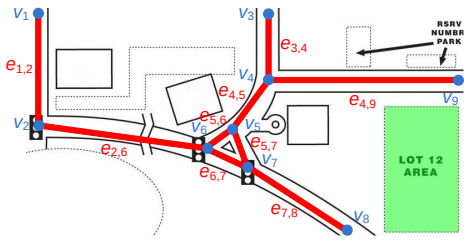


Fig. 1. Campus parking lot's map

The remainder of this paper is organized as follows. Section II describes the system model and the CAEM problem. Section III proposes our design to solve the problem. Section IV evaluates the performance of our proposed method in comparison with other methods. Section V presents related work. Section VI concludes this paper with remarks on our future work.

II. Problem Statement

In this section, we will describe the new CAEM problem that has not been studied before. Its goal is to determine all vehicles' velocity profiles to minimize their total energy consumption while avoiding the collision among the vehicles in single lanes. Before describing the problem, we will first introduce the system model (including the *vehicle traffic model* and the *vehicle energy consumption model*) in Section II-A. We will then list all the constraints for vehicles' mobility and formally formulate the CAEM problem in Section II-B.

A. System Model

Vehicle traffic model. First, we consider a scenario where K *vehicles* are running on the roads in the region of a vehicle transportation system, which can be modeled as a *map* $G = (P,E)$, where $P = \{p_1, ..., p_N\}$ denote the *junction set* and $E \subseteq P \times P$ denote the *route segment set*. A *route segment* on a map is defined as a route with no traffic light, stop sign, or any other route connected to it except its two end junctions. We call the end junctions the *vertices* of the route segment. A route segment is denoted by $e_{i,j}$ if its two vertices are p_i and p_j. For example, Fig. 1 shows the map of a campus's parking lot, where there are 9 junctions $(p_1, ..., p_9)$ (blue points) and 9 segments ($e_{1,2}$, $e_{2,6}$, ..., $e_{7,8}$) (red lines). Notice that a junction can be the vertex of multiple route segments. For example, p_5 is the vertex of three segments $e_{4,5}$, $e_{5,6}$, and $e_{5,7}$. Suppose that p_{k_0}, p_{k_1}, p_{k_2}, ..., $p_{k_{n_k}-1}$, $p_{k_{n_k}}$ are junctions that vehicle k travels through (where p_{k_0} and $p_{k_{n_k}}$ are the source and the destination). Then, we can use a vector, called *route vector*:

$$\mathbf{e}_k = \left[e_{k_0,k_1}, e_{k_1,k_2}, ..., e_{k_{n_k-1},k_{n_k}}\right] \quad (1)$$

to represent the sequence of the route segments that vehicle k travels through.

We consider a discrete time system where time $t = 1, 2, ...$ and the time constraint for each vehicle k is T_k, which is the maximum time that vehicle k can accept to drive from its source to its destination. In addition, we use $l_{i,j}$, $v_{i,j}^{\max}$, and $v_{i,j}^{\min}$ to represent the length, the maximum speed, and the minimum speed of route segment $e_{i,j}$, respectively. We use $v_{k,t}$, $a_{k,t}$, and $d_{k,t}$ to represent the velocity, the acceleration, and the traveling distance of vehicle k at time t, respectively. Here, $a_{k,t} = v_{k,t} - v_{k,t-1}$. Finally, for each vehicle k, we use t_{k_0} to represent its starting time to join the system.

Vehicle energy consumption model. In this paper, we take the fuel energy consumption model as an example. To apply our method to the electric vehicles, we only need to replace this model with the energy consumption model for electric vehicles [3]. We use a widely used vehicle energy consumption model, called *comprehensive modal emission model* [5], [6], to describe the energy consumption of vehicle k at time t:

$$J(v_{k,t}, a_{k,t}) = \frac{1}{44}\left(\frac{A_3 v_{k,t}^3 + A_2 v_{k,t}^2 + A_1 v_{k,t} + M a_{k,t}}{0.4\eta_{\text{tf}}} + V\right) \quad (2)$$

where A_1 means the air-drag factor, A_2 means speed-correction coefficient to rolling resistance, A_3 means the rolling resistance coefficient, g means the gravitational constant (9.81m/s^2), η_{tf} means the combined efficiency of the transmission, α means the engine friction factor, β means the engine speed (revolutions per second), and V means the engine displacement (liter). For simplicity, we can represent Equ. (2) by:

$$J(v_{k,t}, a_{k,t}) = B_3 v_{k,t}^3 + B_2 v_{k,t}^2 + B_1 v_{k,t} + B_0 + B a_{k,t} \quad (3)$$

where $B = \frac{M}{17.6\eta_{\text{tf}}}$, $B_0 = \frac{V}{44}$, $B_i = \frac{A_i}{17.6\eta_{\text{tf}}}$ $(i = 1, 2, 3)$.

B. Problem Statement

Before formulating the problem, we first list the mobility constraints of each vehicle as follows:

1) Speed limit constraint. Given a vehicle k and its driving route segment $e_{k_i,k_{i+1}}$, the vehicle's velocity $v_{k,t}$ needs to satisfy the speed limit of $e_{k_i,k_{i+1}}$

$$v_{k_i,k_{i+1}}^{\min} \le v_{k,t} \le v_{k_i,k_{i+1}}^{\max}. \quad (4)$$

2) Drivers' comfort constraint. For drivers' comfort, the acceleration of the vehicle cannot be too large. Hence, the following limitations are imposed on the vehicles' acceleration

$$a_{\min} \le a_{k,t} \le a_{\max}, \ \forall t = 0, 1, 2, ... \quad (5)$$

where $a_{\min}$ and $a_{\max}$ are -1.5m/s^2 and $2.5\ \text{m/s}^2$ [13]. Notice that when acceleration is negative, it means the velocity is decreased.

3) Stop sign constraint. Each vehicle must stop in front of the stop sign at t when its velocity is not zero at $t-1$:

$$v_{k,t} = 0, \text{ if } v_{k,t-1} \neq 0 \text{ and } d_{k,t} \in \mathscr{D}_{\text{stop}} \quad (6)$$

where $\mathscr{D}_{\text{stop}}$ denotes the set of distances of the stop signs from the source in the vehicle k's route.

4) Traffic light constraint. Suppose there are L_k traffic lights in the route of vehicle k and let $d_{k,1}^{\text{sig}}, d_{k,2}^{\text{sig}}, ..., d_{k,L_k}^{\text{sig}}$ denote their distance from the source. Then, when $d_{k,t} = d_{k,l}^{\text{sig}}$, i.e., the vehicle is located at the l^{th} traffic light, we have the constraint

$$v_{k,t} = 0, \text{ when } t \in \mathscr{T}_{k,l}^{\text{red}}, \quad (7)$$

where $\mathscr{T}_{k,l}^{\text{red}}$ denotes the time intervals that the lth traffic light is red.

5) Vehicle influence constraint. If two vehicles drive on a route segment on a single lane, then the vehicle behind, say vehicle i, cannot exceed the previous vehicle, say vehicle j. Let t_i and t_j denote the time points that vehicle i and j arrive at the route segment. Then, $d_{i,t} - d_{i,t_i}$ and $d_{j,t} - d_{j,t_j}$ respectively represent the distance that vehicle i and vehicle j have traveled the route segment at time t. Accordingly, we have the following constraint in this route segment:

$$d_{i,t} - d_{i,t_i} < d_{j,t} - d_{j,t_j}, \forall t \quad (8)$$

if $t_i > t_j$, i.e., vehicle j arrives at the route segment before vehicle i.

Collision-Aware vehicle Energy consumption Minimization (CAEM): The objective of CAEM is to find the optimal velocity profile for all the vehicles on the roads in the vehicle transportation system for a given region to minimize the overall energy consumption, while satisfying the above five constraints. Then, the CAEM problem can be mathematically represented as

$$\min \quad \sum_{k=1}^{K} \sum_{t=t_{k_0}}^{t_{k_0}+T_k} J(v_{k,t}, a_{k,t}) \quad (9)$$

$$\text{s.t.} \quad \text{Constraints (4) - (8) are satisfied,} \quad (10)$$

and the output of this problem is the velocity profile of each vehicle: $\{v_{k,t_{k_0}}, v_{k,t_{k_0}+1}, ..., v_{k,t_{k_0}+T_k}\}$, $(k = 1, ..., K)$.

Solving this formulated problem is non-trivial. One solution to this CAEM problem is that after the cloud receives the information from all vehicles, it calculates the optimal velocity profile of each vehicle. However, this solution is not effective due to the three challenges indicated in Section I. First, since the problem considers the traffic light constraint and vehicle influence constraint formulated by Equ. (7) and Equ. (8), both of which are non-convex [8], we cannot obtain the optimal velocity profile using the existing optimization methods [8]. Second, it is impractical to collect the information of all vehicles at the beginning. Third, calculating the velocity profiles based on the information of all vehicles is not efficient. In Section III, we introduce our solution for CAEM called *CAEM-S* that can tackle the three challenges.

III. System Design

As introduced in Section I, to tackle the challenge 2 and challenge 3, we calculate vehicles' velocity profiles in sequence. That is, every time after receiving a vehicle's request, the cloud calculates this vehicle's optimal velocity profile that avoids collision with existing vehicles. Fig. 2 shows the

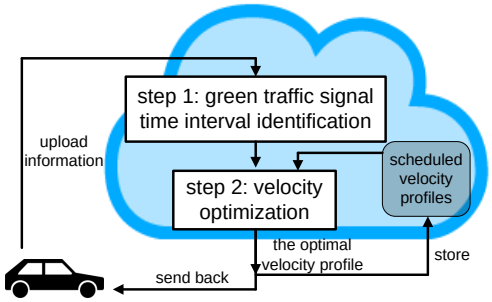


Fig. 2. The architecture of CAEM-S.

architecture of our solution. The cloud maintains a table, called the *scheduled velocity table* (or simply *SV-Table*), to record the velocity profiles of the previously scheduled vehicles. When a vehicle, say k, starts its travel, it uploads its request including the information of its starting time, travel time constraint, and route, to cloud. Based on SV-Table, cloud calculates the optimal velocity profile of vehicle k that also prevents it from colliding with the previously scheduled vehicles. After deriving the velocity, the cloud sends the calculated velocity profile back to vehicle k and stores it in SV-Table. When vehicle k finishes its travel, cloud removes its velocity profile from SV-Table. In this way, the computation of each vehicle's optimal velocity profile that presents it from colliding with other vehicles does not need the inputs of all the vehicles at the beginning, which handles the second challenge. CAEM-S also handles the third challenge by alleviating the computation burden, since each time CAEM-S only needs to calculate one vehicle's velocity.

To handle the first challenge, we transform CAEM from a non-convex problem to a convex problem. We first theoretically prove that reducing the energy consumption of each vehicle is essentially keeping the velocity of the vehicle constant, which is called *constant velocity principle* (Section III-A). According to this principle, CAEM-S keeps every vehicle's velocity constant within each route segment and reduces the velocity difference among different route segments. For the latter objective, CAEM-S identifies the green traffic signal time interval of each traffic light in the source-destination route of a vehicle, during which the vehicle should drive through the traffic light. It means that when a vehicle arrives at the traffic light during the determined time interval, it can directly pass the traffic light without stopping since the traffic signal is green during this period. To identify the green traffic signal time intervals and avoid collision when calculating a vehicle's velocity profile, we propose a novel method called *light schematic map (LSMap)*, which is built on a two dimension (temporal and spatial dimensions) space. After we change the traffic light constraint and vehicle influence constraint from nonlinear to linear, the CAEM problem is simplified to a convex problem, and it can be efficiently solved by the existing optimization methods [8]. In summary, CAEM-S has the following two steps:

Step 1: Green traffic signal time interval identification (Section III-B). Given the locations of all the traffic lights on a vehicle's route and each traffic signal timing schedule, which are provided by cloud storage, CAEM-S needs to find the sequence of green traffic signal time intervals that the vehicle needs to arrive at each traffic light in its route. The objective is to make the change of the vehicle velocity as small as possible.

Step 2: Velocity optimization (Section III-C): Given the identified traffic signal time intervals and the velocity profile of the previously scheduled vehicles, CAEM-S derives the velocity profile of the vehicle, where the objective is to minimize the energy consumption while avoiding collisions.

In the following part, we will first introduce the constant velocity principle and then describe the above steps in detail.

A. Constant Velocity Principle

In this part, we theoretically derive the constant velocity principle in Theorem 3.1. We first give Lemma 3.1 for the preparation of the proof of Theorem 3.1:

Lemma 3.1: (Power means inequality) [19] For any two integers p and q s.t. $p < q$, the following inequality holds:

$$\left(\sum_{t=t_{k_0}}^{T_k+t_{k_0}} w_t v_{k,t}^p\right)^{\frac{1}{p}} \leq \left(\sum_{t=t_{k_0}}^{T_k+t_{k_0}} w_t v_{k,t}^q\right)^{\frac{1}{q}} \tag{11}$$

where $\sum_{t=t_{k_0}}^{T+t_{k_0}} w_t = 1$.

Theorem 3.1: (Constant velocity principle) given the time constraint T_k and the travel distance D_k of vehicle k, its total energy consumption is minimized when the vehicle's velocity is constant, i.e., $v_{k,t} = D_k/T_k$ at each time point.

Proof First look at Equ. (11). Let $p = 1$ and $w_t = \frac{1}{T}$, then according to Lemma 3.1, we obtain

$$\sum_{t=t_{k_0}}^{T_k+t_{k_0}} \frac{v_{k,t}}{T_k} \leq \left(\sum_{t=t_{k_0}}^{T_k+t_{k_0}} \frac{v_{k,t}^q}{T_k}\right)^{\frac{1}{q}} \tag{12}$$

$$\Rightarrow \quad \left(\sum_{t=t_{k_0}}^{T_k+t_{k_0}} \frac{v_{k,t}}{T_k}\right)^{q} \leq \sum_{t=t_{k_0}}^{T_k+t_{k_0}} \frac{v_{k,t}^q}{T_k} \tag{13}$$

Let $\bar{v}_k = \sum_{t=t_{k_0}}^{T+t_{k_0}} v_{k,t}/T_k$. The total energy consumption of vehicle k is calculated by

$$\begin{aligned} &\sum_{t=t_{k_0}}^{T_k+t_{k_0}} J(v_{k,t}, a_{k,t}) \\ =\ &\sum_{t=t_{k_0}}^{T_k+t_{k_0}} \left(B_3 v_{k,t}^3 + B_2 v_{k,t}^2 + B_1 v_{k,t} + B_0 + B a_{k,t}\right) \\ =\ &B_3 \sum_{t=t_{k_0}}^{T_k+t_{k_0}} v_{k,t}^3 + B_2 \sum_{t=t_{k_0}}^{T_k+t_{k_0}} v_{k,t}^2 + B_1 \sum_{t=t_{k_0}}^{T_k+t_{k_0}} v_{k,t} + B_0 n_k + B \sum_{t=t_{k_0}}^{T_k+t_{k_0}} a_{k,t} \end{aligned} \tag{14}$$

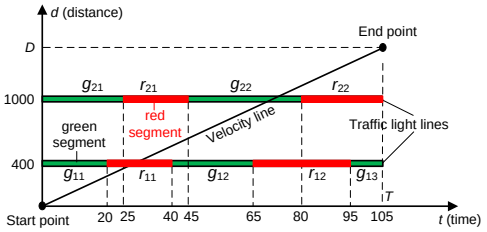


Fig. 3. Light scheduling map (LSMap).

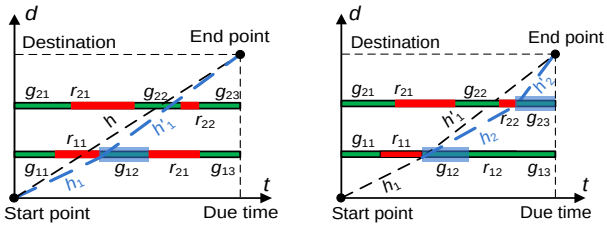


Fig. 4. Green traffic signal time interval identification.

According to Equ. (12), we have $\sum_{t=t_{k_0}}^{T_k+t_{k_0}} v_{k,t}^q \geq T_k \bar{v}_k^q$ $(q = 1,2,3)$. Also, since $a_{k,t} \geq 0 \ \forall k,t$, from Equ. (14), we derive that

$$\sum_{t=t_{k_0}}^{T_k+t_{k_0}} J(v_{k,t}, a_{k,t}) \tag{15}$$

$$\geq B_3 T_k \bar{v}_k^3 + B_2 T_k \bar{v}_k^2 + B_1 T_k \bar{v}_k + B_0 T_k + B \sum_{t=t_{k_0}}^{T_k+t_{k_0}} a_{k,t} \tag{16}$$

$$\geq \sum_{t=t_{k_0}}^{T_k+t_{k_0}} J(\bar{v}_k, 0) \tag{17}$$

which indicates that the energy consumption of vehicle k is minimized when its velocity is constant.

According to the constant velocity principle, to minimize the energy consumption of a vehicle is essentially to keep the vehicle's velocity unchanged. Though maintaining a constant velocity within a route segment is easy, it is impossible to guarantee the velocity of vehicle in different route segments to be the same due to traffic lights at junctions and different speed limits in different route segments. In the next section, we try to reduce the difference between the velocities of different route segments with the aid of LSMap.

B. Green Traffic Signal Time Interval Identification

Before describing our green traffic signal time interval selection algorithm, we first introduce LSMap, which is used to schedule the velocity of vehicles to avoid encountering red traffic signals.

For each vehicle k's request, the cloud builds an LSMap in a temporal-spatial space, where we use t and d to represent the temporal and spatial coordinates, as shown in Fig. 3. A (x,y) point in the LSMap means that the vehicle is y distance from its source at time x. Each traffic light in the vehicle's source-destination route is represented by a *traffic light line* that is parallel to the t axis, where its d coordinate equals the distance between the traffic light and the vehicle's source. Each traffic line is further partitioned to a set of red segments and a set of green segments, representing the green traffic signal time intervals and red traffic signal time intervals of the traffic light, respectively. Finally, a vehicle's start and final status (i.e., time and location) in LSMap are represented by two points $(t_{k_0}, 0)$ and $(t_{k_0} + T_k, D_k)$, namely the *start point* and the *end point*. Then, finding a velocity profile of the vehicle is equivalent to finding a curve to connect the start point and the end point in LSMap, without intersecting with any red segment. Here, we call the curve the *velocity line* of the vehicle.

Fig. 4 gives an example of LSMap. Vehicle k starts at time 0 second, with traveling distance being 1000 meters and time constraint being 105 seconds. There are two traffic lights in vehicle k's route, which are 400 and 1000 meters away from the source, represented by the two traffic light lines with d coordinates 400 and 1000 in LSMap. Take the second traffic light as an example, it has two red traffic signal time intervals ($[25,45]$ and $[80,105]$) and two green traffic signal time intervals ($[0,25]$ and $[45,80]$). Here, r_{li} (g_{li}) (both l and i) represent the ith red (green) traffic signal time interval of lth traffic light.

According to the constant velocity principle, minimizing the energy consumption of a vehicle is essentially to minimize the change of vehicles' velocity. Hence, the ideal case is that the velocity line is a straight line connecting the start point and the end point in LSMap, without intersecting any red segment, which, unfortunately, seldom happens in reality. As shown in Fig. 4(a), the straight velocity line h intersects with a red traffic signal time intervals r_{11}. In addition, the traffic line cannot intersect the green segment if its time interval conflicts with the constraint to avoid vehicle collision (we will present this constraint in Section III-C).

Therefore, we first need to determine the set of green segments that the velocity line intersects to minimize the deviation of the velocity line from the straight velocity line. Here, we propose a time efficient algorithm for the green segment selection. Its basic idea is to select the green segment for the lowest traffic light line to the highest traffic light line. For each traffic light line, we check whether the intersection point (denoted by p) with the velocity line is in a red segment or an occupied green segment. If yes, we select the nearest unoccupied green segment to point p. Otherwise, we select the green segment that p is located at. Fig. 4(a) gives an example. The velocity line has junctions with r_{11} and r_{22}. Then, we need to adjust the velocity line h to the broken line $\{h_1, h_1'\}$, where h_1' intersects with an occupied green segment g_{22}. Then, in the second step, as shown in Fig. 4(b), we adjust the line h_1' to the broken line $\{h_2, h_2'\}$. Finally, the velocity line of the vehicle is $\{h_1, h_2, h_2'\}$. Here, in each iteration l, we use h_1, ..., h_l to represent the velocity lines from the source to the lth route segments, and use h_l' to represent the unadjusted velocity line from the $l-1$th route segment to the destination.

For the identified green segment $[t_{\text{sta}}^l, t_{\text{end}}^l]$ for each traffic

light l in vehicle k's source-destination route, it has the following constraint when arriving at traffic light l, called *green segment constraint*:

$$t^l_{\text{sta}} \le t \le t^l_{\text{end}}, \text{ when } d_t = d^l_{\text{sig}}. \tag{18}$$

C. Velocity Optimization

In the previous section, we introduced how to determine the range of velocity profile of a vehicle to avoid red lights in its route. In this section, we introduce how to finalize the velocity profile of the vehicle that avoids collision with existing vehicles.

Avoiding Collision. First, we discuss in which cases two vehicles will collide with each other, i.e., there exists an intersection of the vehicles' velocity lines, in a given route segment, say $e_{i,j}$. We denote the arriving time and the leaving time of vehicle k (vehicle l) for $e_{i,j}$ by t^{s}_k and t^{e}_k (t^{s}_l and t^{e}_l). Now consider the following four cases:

- As shown in Fig. 5(a), when $t^{\text{s}}_k < t^{\text{s}}_l$ and $t^{\text{e}}_k < t^{\text{e}}_l$ (case 1), or $t^{\text{s}}_k > t^{\text{s}}_l$ and $t^{\text{e}}_k > t^{\text{e}}_l$ (case 2): the new velocity has no intersection with the existing velocity line.
- As Fig. 5(b) shows, when $t^{\text{s}}_k < t^{\text{s}}_l$ and $t^{\text{e}}_k > t^{\text{e}}_l$ (case 3), or $t^{\text{s}}_k > t^{\text{s}}_l$ and $t^{\text{e}}_k < t^{\text{e}}_l$ (case 4): the new velocity line has intersection with the existing velocity line.

Accordingly, we obtain the following constraint for vehicle k to avoid colliding with vehicle l:

$$t^{\text{e}}_k \begin{cases} < t^{\text{e}}_l & \text{if } t^{\text{s}}_k < t^{\text{s}}_l \\ > t^{\text{e}}_l & \text{if } t^{\text{s}}_k > t^{\text{s}}_l \end{cases} \tag{19}$$

Hence, suppose that vehicle l's velocity profile has been determined, to calculate the velocity profile of vehicle k, which enters the system later than vehicle l, we only need to judge whether the starting time point of vehicle k is larger than the starting time point of vehicle l. If yes, then the leaving time of vehicle k should be also larger than that of vehicle l; otherwise, the leaving time of vehicle k should be smaller than that of vehicle l.

If we need to schedule the newly added vehicle k, for each route segment in vehicle k's route, say $e_{i,j}$, then we need to check all the previously scheduled vehicles in $e_{i,j}$. Suppose that there have been n vehicles $l_1, l_2, ..., l_n$ scheduled in $e_{i,j}$. Let $\mathscr{L}_1$ and $\mathscr{L}_2$ respectively represent the set of vehicles arriving at $e_{i,j}$ before and after vehicle k. Then, the leaving time of vehicle k should satisfy the following constraint, called the *simplified vehicle influence constraint*:

$$t^{\text{e}}_k \begin{cases} > \max_{l_i \in \mathscr{L}_1} \{t^{\text{e}}_{l_i}\} \\ < \min_{l_i \in \mathscr{L}_2} \{t^{\text{e}}_{l_i}\} \end{cases} \tag{20}$$

It means vehicle k's leaving time from the road segment must be later than the latest leaving time of the vehicles that arrive at the road segment before it, and earlier than the earliest leaving time of the vehicles that arrive at the road segment after it.

Therefore, when determining green segments for a vehicle k in Section III-B, green segment $[t^l_{\text{sta}}, t^l_{\text{end}}]$ cannot be selected if there is no time point in $[t^l_{\text{sta}}, t^l_{\text{end}}]$ satisfying Equ. (20), i.e., if either of the following two happens: 1) $t^l_{\text{end}} < \min_{l_i \in \mathscr{L}_1} \{t^{\text{e}}_{l_i}\}$ or 2) $t^l_{\text{sta}} > \max_{l_i \in \mathscr{L}_2} \{t^{\text{e}}_{l_i}\}$.

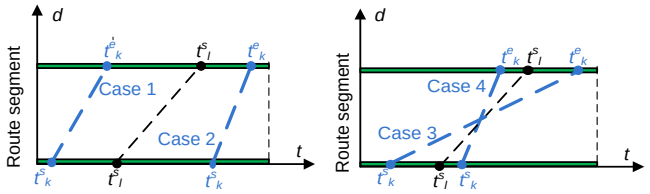


Fig. 5. Four cases for two velocity lines.

SV-Table. According to Equ. (20), to obtain the simplified vehicle influence constraint, it is required to know all the previously scheduled vehicles' arriving time and leaving time for each route segment. Here, CAEM-S stores this information in SV-Table. As Fig. 6 shows, for each route segment $e_{i,j}$, SV-Table has an entry including the starting time and ending time of all the scheduled vehicles' $\{l_1, l_2, ..., l_n\}$ that travel through $e_{i,j}$. That is, it provides the information required in the simplified vehicle influence constraints (Equ. (20)). When there is new vehicle scheduled, CAEM-S adds the vehicle's arriving time and leaving time for all the route segments to the corresponding entries in SV-Table. On the other hand, if a vehicle leaves the system, CAEM-S will remove its arriving time and leaving from all the entries in the SV-Table.

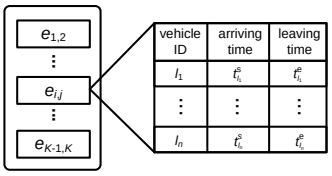


Fig. 6. Scheduled velocity table (SV-Table).

Simplified Non-convex Problem. Now, what remains to be done is to replace the two non-convex constraints, i.e., Equ (7) and Equ. (8), by the green segment constraint (Equ. (18)) and the vehicle mobility constraint (Equ. (20)), both of which are simply linear constraints. Then, we formulate the following optimization problem for vehicle k:

$$\min \sum_{t=t_{k_0}}^{t_{k_0}+T_k} J(v_{k,t}, a_{k,t}) \tag{21}$$

$$\text{s.t. Equ. (4), (5), (18), (20) are satisfied}$$

of which the objective is to minimize the energy consumption of vehicle k, i.e., $\sum_{t=t_{k_0}}^{t_{k_0}+T_k} J(v_{k,t}, a_{k,t})$ and the output of the problem is the velocity profile of each vehicle. The feasible region of this problem is linear, which can be directly solved using the existing optimization methods. Let n_k represent the number of route segments that vehicle k travels through, then the time complexity for each vehicle k is $O(n_k^2)$ if we use the subgradient method [8].

IV. Performance Evaluation

In this section, we compared the performance of CAEM-S with two velocity optimization methods, called dynamic

programming (DP) [13] and predictive cruise control (PCC) [2]. Similar to our approach, in DP, each vehicle uploads its current coordinate and velocity to the cloud, which uses DP to derive the vehicle's velocity profile. More specifically, each vehicle's status can be represented by a 2-dimensional point (distance, velocity), which indicates the velocity when the vehicle arrives at the place with the indicated distance from its source. Then, DP builds a graph that enumerates all possible paths from the source point to the destination point, and iteratively calculates the energy consumption to reach each point from the original point in the graph. After deriving the cost of the destination point, DP can obtain the optimal velocity profile by backtracking the destination point. In PCC, each vehicle contacts the upcoming traffic light to get the traffic light information, i.e., the location and signal timing, and calculates the optimal velocity profile to avoid stopping at the upcoming red traffic light. Both DP and PCC determine each vehicle's velocity profile without trying to avoid vehicle collision. Further, DP does not consider the effect of traffic lights, and hence DP cannot prevent vehicles stopping in front of red traffic signals. PCC cannot provide the optimal velocity profile when a vehicle is outside of the transmission range of a traffic light.

We used both trace-driven simulation (Section IV-A) and real-world testbed experiment (Section IV-B) to compare the performance of CAEM-S with DP and CPP. The metrics we measured include: *the total number of velocity violations*, defined as the number of times that vehicles violate the suggested velocity, *the total energy consumption*, and *the average computation delay*. Here, we do not distinguish the computation capacity of cloud and the traffic lights. In the real-world testbed experiment, computation delay also includes the time duration that vehicle upload their information to the server.

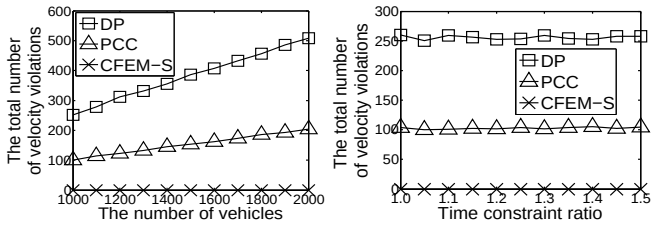
A. Trace-driven Simulation

We conducted simulation using MATLAB based on the real vehicle mobility trace from the Cologne urban area, which covers a region of 400 square kilometers with 404 traffic lights. The trace records the locations of more than 700,000 individual vehicle trips [1] for a time period of 24 hours. The information includes the destination of each taxi, and its timestamp, vehicle ID, GPS coordinate, and velocity every minute. We randomly picked up a number of vehicles (the number is changed from 1000 to 2000). Also, we changed the time constraint of each vehicle (to arrive at its destination) to test how time constraint affects the performance. More specifically, we defined a metric, called the *time constraint ratio* for each vehicle as follows:

$$\text{time constraint ratio} = \frac{\text{time constraint}}{\text{actual traveling time in the trace}}$$

For example, if the time constraint ratio is set to be 1.5, then the time constraint for each vehicles is set to be 1.5 times of the time that the vehicle travels in the trace.

1) The Total Number of Velocity Violations: We first compared the total number of velocity violations of CAEM-S, DP, and PCC with different number of vehicles and different time constraint ratio in Fig. 7(a) and Fig. 7(b), respectively. From both figures, we find that the total number of velocity violations follows: DP $>$ PCC $>$ CAEM-S. Recall that both DP and PCC consider each vehicle as an independent object without considering the possible collisions between in the vehicles in single lanes, hence the vehicles in DP and PCC are unable to follow the suggested velocity all the time. In particular, DP has the highest number of violations, as DP also does not consider the effect of red traffic signals on vehicles' mobility and hence it has to violate the suggested velocity when confronting red signals. We also observe that the number of violations increases with the increase of the number of vehicles in DP and PCC. It is because more vehicles in the region lead to more vehicles in a route segment, which generates higher likelihood of vehicle collisions.



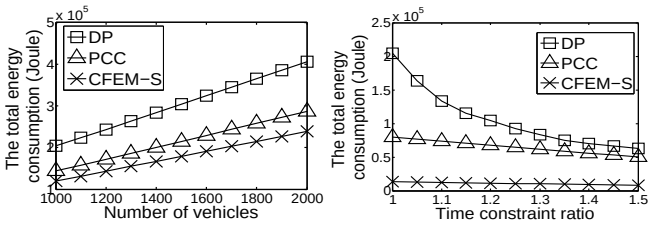
(a) Different # of vehicles (time constraint ratio = 1.0)

(b) Different time constraint ratio (# of vehicles = 1000)

Fig. 7. The total number of velocity violations in the roads with single lanes (simulation).

2) The Total Energy Consumption: We then compared vehicles' total energy consumption of DP, PCC, and CAEM-S in Fig. 8(a) and Fig. 8(b), with different number of vehicles and different time constraint ratios, respectively. From both figures, we find that the total energy consumption follows: PCC $>$ DP $>$ CAEM-S. As we mentioned before, both DP and PCC do not consider the possible collisions among the vehicles in single lanes, which makes the vehicles unable to follow the suggested velocity in reality (as shown in Fig. 7(a)(b)). Different from PCC and DP, CAEM-S takes into account the collisions among vehicles in single lanes, and hence vehicles can accurately follow the velocity profiles in CAEM-S, leading to lower energy consumption than both DP and PCC. In addition, PCC has higher energy consumption than CAEM-S because PCC only lets vehicle adjust the velocity when it moves to the upcoming traffic light's transmission range and hence can only achieve a local optimal solution. On the other hand, DP has higher energy consumption than both CAEM-S and PCC since 1) DP does not take into account the effect of traffic light on vehicles' velocity profiles, leading to more stops and accelerations in front of traffic lights, and hence more energy consumed, and 2) DP has higher computation delay, which causes the vehicles unable to obtain the optimal velocity from the cloud in time, and hence hinders the vehicles to follow the suggested velocity closely.

From Fig. 8(b), we also find that with the increase of time constraint ratio, the total energy consumption in all the three methods decreases. It is because that when the time constraint



(a) Different # of vehicles (time constraint ratio = 1.0)

(b) Different time constraint ratio (# of vehicles = 1000)

Fig. 8. The total energy consumption (simulation).

of each vehicle is larger, the average velocity for each vehicle is smaller, which leads to lower energy consumption according to the vehicle energy consumption model (Equ. (2)).

B. Real implementation

In the real-world testbed experiments, we built an Android application for mobile phones, and equipped the three vehicles with the mobile phones, where the Android phones only serve the communication between vehicles and the server. We labeled the three vehicles by vehicle 1, vehicle 2, and vehicle 3, and drove these vehicles around a university campus to test the performance of DP, PCC, and CAEM-S, where vehicle 1 enters the route first, and then vehicle 2, and finally vehicle 3. When vehicles are driving on the road, each Android phone records the actual velocity of its vehicle in every 2 seconds. This route has a length of 2000 meters, three traffic lights (located at 108 meters, 1234 meters, and 1878 meters from the source). The time constraint was set to be 2 minutes. We block the place names to follow the conference's double blind policy. We used a server in a super computing center in our campus as the cloud, where the server has 8GB memory and Intel core i3 processor series.

In all the strategies, before the vehicles started, the Android phones uploaded their vehicles' information to the server and the server sent back the velocity profile after it derived the vehicles' velocity profiles. There are two cases when vehicles possibly cannot follow the suggested velocity profile: (a) there will be a collision if the vehicle still follows the velocity profile and (b) the upcoming traffic signal is red but no deceleration is suggested in the velocity profile. Once a vehicle does not follow the suggested velocity, it needs to upload its information to the server again, and the sever will re-calculate the velocity profile.

We compared the total number of velocity violations of CAEM-S, DP, and PCC in Fig. 9. From the figure, we find that the total number of velocity violations follows: DP > PCC > CAEM-S, which is consistent with the result in Fig. 7. To better observe how vehicles followed the suggested velocity profiles, we

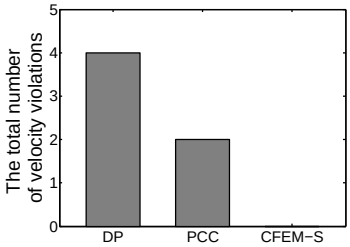


Fig. 9. The total number of velocity violations (real-world).

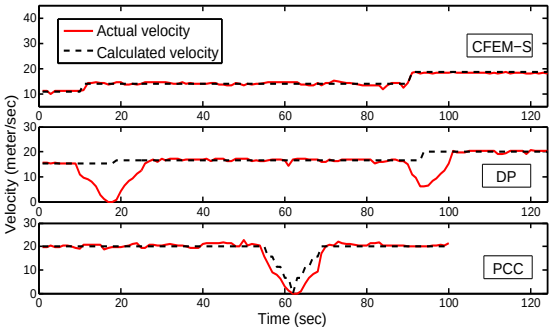


Fig. 10. Actual velocity vs. suggested velocity.

picked one vehicle, say vehicle 2, and compared the suggested velocity profile and the actual velocity profile for this vehicle under DP, PCC, CAEM-S in Fig. 10. The experimental results confirm that the vehicle cannot always follow the suggested velocity profiles in practice. As we expected, CAEM-S has fewer accelerations compared with DP and PCC and hence generates less energy consumption. In particular, DP has two accelerations at the 19th second and the 87th second, respectively, and PCC has one acceleration at the 53rd second.

V. Related Work

A. Local vehicle velocity optimization

During recent years, many efforts have been devoted to reducing energy consumption through optimizing vehicles' velocity profiles. However, to calculate the optimal velocity profile for each vehicle from its source to its destination, it requires large storage and intensive computations, which is beyond the capability of vehicles' devices. Hence, many strategies select to obtain local traffic information from roadside units (RSUs) or traffic lights and let vehicles calculate a local optimal velocity profile for vehicles [2], [18]. For example, Asadi and Vahidi [2] proposed a control algorithm that adapts the velocity profile to guarantee that a vehicle approaches a traffic light at green whenever possible. The authors used a short-range radar and traffic light information to predictively schedule a suboptimal velocity profile and implemented the algorithm in an existing cruise control system. A similar approach was proposed by Raubitschek *et al.* [18], in which they divided the velocity profile into a number of modes (e.g., accelerating, decelerating, and stop) and generated a velocity profile combined with these modes to ensure arrivals at a green traffic light. In [9] and [7], Ivarson *et al.* proposed a velocity profile optimizing algorithm for a certain look-ahead distance. The authors used a short-range radar and traffic signal information to predictively schedule a suboptimal velocity trajectory and implemented the algorithm in an existing cruise control system.

B. Global vehicle velocity optimization

The above approaches can only generate suboptimal solutions for the entire trip distance because the combination of the optimal solutions of all the route segments does not equal to the optimal solution of the entire trip. Recently, a computing framework, called *remote computing*, has been

proposed to augment the mobile systems' capabilities by migrating computation to more resourceful computers in the cloud [4], [12], [16], [17]. This framework has also been applied in transportation systems, named vehicular cloud [13], [14].

For example, in [13], Ozatay *et al.* let each vehicle upload its information, e.g., velocity and location, to the cloud through base stations, and based on the uploaded information and the associated global traffic regulator information stored in the cloud, the cloud then derives the optimal velocity profile for the vehicle using the dynamic programming (DP). In [14], Ozatay *et al.* also presented a non-linear velocity calculation problem with several user defined constrains (e.g., driver comfort constraint) and they developed an analytical solution to generate an optimal velocity profile to minimize energy consumption on a given route with the existence of a set of traffic lights. Similarly in [14], a closed-form solution is proposed for the generation of optimal energy management in electric vehicles for a given route. Our method is advantageous than the previous methods in that our method considers the possible collisions among vehicles in single lanes, and hence helps vehicles better follow the suggested velocity profiles, which consequently reduces the energy consumption.

VI. Conclusion

For the vehicle velocity optimization problem, it is critical to consider the influence among the vehicles in the same lane in order to avoid vehicle collision, which however is neglected in the previous proposed methods. Accordingly, in this paper, we formulated a new problem, called the CAEM problem, of which the objective is to minimize the energy consumption of all the vehicles while avoiding the collisions between the vehicles. CAEM is non-trivial to solve because it needs to consider all the vehicles together while vehicles may join and leave the region at any time. As a solution, we proposed an online algorithm called the Serial Energy-efficient Vehicle Scheduling (CAEM-S) to calculate vehicles' velocity profiles in a serial fashion, which decreases the time complexity and only requires the information of each existing vehicle. To further increase the time efficiency of CAEM-S, we simplify the constraints in CAEM-S from nonlinear to linear. In particular, CAEM-S is composed of two steps: 1) *Green traffic signal time interval identification*, which transfers the traffic light constraint to be linear, and 2) *Velocity optimization*, which transfers the vehicle mobility constraint to be linear, and hence we can obtain the optimal velocity using the existing optimization method.

Finally, we conducted extensive trace-driven simulation and real-world testbed experiment to compare CAEM-S with two state-of-the-art methods, DP and PCC. The experimental results demonstrate the superiority of CAEM-S over the previous methods in terms of the number of velocity violations, the total energy consumption and the computation time. In our future work, we will further improve the velocity optimization system in the following aspects: 1) Actually, vehicles also cannot pass the previous vehicles in the road with multiple lanes in some cases, especially when the traffic is heavy. So, we will also consider the collisions among vehicles in the route segments with multiple lanes. 2) In reality, some vehicles are not willing to follow the suggested velocity from the cloud. Hence, we will take into account the how to avoid the collisions with these human-determined vehicles.

VII. Acknowledgements

This research was supported in part by U.S. NSF grants OAC-1724845, ACI-1719397 and CNS-1733596, and Microsoft Research Faculty Fellowship 8300751.

References

[1] Vehicular mobility trace of the city of Cologne. http://kolntrace.project.citi-lab.fr/. [Accessed: March 2017].

[2] B. Asadi and A. Vahidi. Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. *Trans. Control Syst. Technol*, 2011.

[3] C. T. Calafate, G. Fortino, S. Fritsch, J. Monteiro, J.-C. Cano, and P. Manzoni. An efficient and robust content delivery solution for IEEE 802.11p vehicular environments. *Journal of Network and Computer Applications*, 35(2):753–762, 2012.

[4] B.-G. Chu, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proc. of Eurosys*, 2011.

[5] W. F. Faris, H. A. Rakha, R. I. Kafafy, M. Idres, and S. Elmoselhy. Vehicle fuel consumption and emission modeling an in-depth literature review. *Int. J. Vehicle Systems Modeling and Testing*, 2011.

[6] E. Hellstrom and M. Ivarson. Average annual emissions and fuel consumption for gasoline-fueled passenger cars and light trucks. *United States Environment Protection Agency*, 2008.

[7] E. Hellstrom, J. A. M. Ivarson, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Eng. Pract*, 2009.

[8] F. S. Hillier. *Linear and Nonlinear Programming*. Stanford University, 2008.

[9] M. Ivarson, J. Aslund, and L. Nielsen. Look-ahead control consequences of a non-linear fuel map on truck fuel consumption. In *Proc. Inst. Mech. Eng. D, J. Autom. Eng.*, 2009.

[10] M. Kamal, M. Mukai, J. Murata, and T. Kawabe. Ecological driver assistance system using model-based anticipation of vehicle-road-traffic information. *IET Intell. Transp. Syst.*, 2010.

[11] H. Khayyam, S. Nahavandi, and S. Davis. Adaptive cruise control lookahead system for energy management of vehicles. *Exp. Syst. Appl.*, 2012.

[12] P. Mohan, S. Nath, and O. Riva. Prefetching mobile ads: Can advertising systems afford it? In *Proc. of Eurosys*, 2013.

[13] E. Ozatay, S. Onori, J. Wollaeger, U. Ozguner, G. Rizzoni, D. Filev, J. Michelini, and S. D. Cairano. Cloud-based velocity profile optimization for everyday driving: A dynamic-programming-based solution. *Transaction on Intelligent Transportation Systems*, 2014.

[14] E. Ozatay, U. Ozguner, S. Onori, and G. Rizzoni. Analytical solution to the minimum fuel consumption optimization problem with the existence of a traffic light. In *Proc. of DSCC*, 2012.

[15] S. Park, K. A. H. Rakha, and K. Moran. Predictive eco-cruise control: Algorithm and potential benefits. In *Proc. Forum Integr. Sustainable Transp. Syst., Vienna, Austria*, 2011.

[16] D. Perkins, N. Agrawal, A. Aranya, C. Yu, Y. Go, H. V. Madhyastha, and C. Ungureanu. Simba: Tunable end-to-end data consistency for mobile apps. In *Proc. of Eurosys*, 2015.

[17] Z. Qian, Y. He, C. Su, Z. Wu, H. Zhu, T. Zhang, L. Zhou, Y. Yu, and Z. Zhang. Timestream: Reliable stream computation in the cloud. In *Proc. of Eurosys*, 2013.

[18] C. Raubitschek, N. Schutze, E. Kozlov, and B. Baker. Predictive driving strategies under urban conditions for reducing fuel consumption based on vehicle environment information. In *Proc. of Forum Integr. Sustainable Transp. Syst.*, 2011.

[19] S. M. Ross. *Introduction to Probability Models, 8th Edition*. Amsterdam: Academic Press, 2003.