Proceedings of the Transportation Research Board (TRB), 2009
Washington, DC, January 10-14, 2009

THIS IS NOT THE FINAL VERSION

Delay Pattern Estimation for Signalized Intersections Using Sampled Travel Times

Xuegang (Jeff) Ban*
Department of Civil and Environmental Engineering
Rensselaer Polytechnic Institute (RPI)
110 Eighth Street, room JEC 4034
Troy, NY 12180-3590
Phone: (518) 276-8043 Fax: (518) 276-4833
Email: banx@rpi.edu

Ryan Herring
Department of Industrial Engineering and Operations Research
University of California, Berkeley
2105 Bancroft Way, Suite 300
Phone: (510) 642-5667 Fax: (510) 642-0910
Email: ryanherring@berkeley.edu

Alexandre M. Bayen
Department of Civil and Environmental Engineering,
Systems Engineering,
University of California, Berkeley
711 Davis Hall
Berkeley, CA 94720-1720
Phone (510) 642-2468 Fax (510) 643-8919
Email: bayen@ce.berkeley.edu

Peng Hao
Department of Civil and Environmental Engineering
Rensselaer Polytechnic Institute (RPI)
110 Eighth Street, room JEC 4034
Troy, NY 12180-3590
Email: haop@rpi.edu

Resubmitted For Presentation and Publication
88th Annual Meeting of Transportation Research Board
November 15, 2008

# of Words: 5,382
+ (1 table and 9 figures) = 2,500
TOTAL: 7,882

** Corresponding Author*

Proceedings of the Transportation Research Board (TRB), 2009
Washington, DC, January 10-14, 2009

Abstract

Intersection delays are the major contributing factor to arterial delays. In this article, we study methods to estimate intersection delay patterns using measured travel times. The delay patterns provide a way to estimate the delay for any vehicle arriving at the intersection at any time, which is very useful for providing time-dependent intersection delay information to the driving public. The model requires sampled travel times between two consecutive locations on arterial streets, most likely upstream and downstream of a signalized intersection, without the need to know signal timing or traffic flow information. Signal phases can actually be estimated from the delay patterns, which is a unique feature of the proposed method in this paper. The proposed model is based on two observations regarding delays for signalized intersections: 1) delay can be approximately represented by piecewise linear curves due to the characteristics of queue forming and discharging before a signalized intersection, and 2) there is a non-trivial increase in delay after the start of the red time, which enables us to detect the start of a cycle. We then develop a least square based estimation algorithm to match measured delays in each cycle using piecewise linear curves. The proposed model and algorithm are tested using field experiment data with reasonable results.

1 Introduction

Travel time or delay (defined as the difference between the experienced travel time and some standard travel time such as the free flow travel time) is one of the most important roadway traffic metrics. This is because (1) travel time is a crucial measure of traffic conditions and system performance; (2) travel time represents information that is easy for the driving public to understand and process; (3) travel time information can arguably enable travelers to make educated choices about their itinerary, departure time or even transportation mode, with the result leading to a form of"system self-management." Providing travel times on freeway routes, e.g. via freeway Changeable Message Signs (CMS), has now become a common practice in many states in the US.

Arterial travel time information however is not widely available. This is due to the difficulty of estimating arterial traffic conditions because arterial traffic is fundamentally different from freeway traffic. The difference in traffic flow patterns is mainly due to the existence of traffic signals, stop signs, and cross traffic that introduce interruptions to arterial traffic flow. These interruptions bring discontinuities to quantities of interest when trying to estimate arterial flow, such as travel times or delays. In addition, distinct from freeways, there are usually many possible routes from an origin to a destination in an arterial network. Providing travel times only for one or a few routes may not be sufficient for a driver to get a full picture of arterial traffic conditions. Therefore, providing time-dependent delay information for arterial intersections seems more desirable. In this article, we focus on signalized intersections as delays before traffic signals are usually the major contributing factor to arterial delays.

In the past, various models have been developed to estimate arterial travel times, with the focus on signalized intersection delays. For example, statistical methods are proposed in [1, 2, 3], in which travel times are modeled as a linear combination of occupancy, flow, and signal parameters. Xie et al. [4] treat arterial link travel time as the summation of cruise time and signal delay. The cruise time is computed using detector speeds and the signal delay is estimated using simplified intersection queuing diagram which requires basic signal parameters (such as cycle length, effective green time, flow/capacity ratio etc). An improved speed-flow relationship is developed in [5] which is shown to be effective to calculate arterial link travel times [6]. The above models are mainly for estimating average (or static) arterial travel times, and recent attention has been focused on estimating dynamic (or time-dependent) arterial travel times [7, 8]. In [7], link travel time is modeled as the summation of free flow travel time and signal delay, while the latter consists of three components: single vehicle delay, queuing delay, and oversturation delay. In particular, the calculation of signal delay requires 30-sec traffic volume and detailed signal timing parameters. By utilizing high-resolution (second-by-second) traffic signal events data (such as phase/timing changes) and vehicle actuation data, "virtual" vehicle trajectories are constructed in [8], which make it possible to estimate accurate dynamic arterial travel times.

As we can see from the above discussion, most existing arterial models require, as a minimum, the knowledge of traffic signal timing parameters and traffic volume to estimate arterial travel times or delays. Collecting traffic signal data for wide-area arterial streets is not trivial since historically traffic signals have been operated and maintained by multiple agencies. On the other hand, by utilizing the vehicle re-identification technique, it has been shown in [9, 10, 11] that samples of intersection delays can be obtained directly. In particular, a new scheme is proposed in [12], in which wireless traffic sensors are deployed downstream (at a fixed distance such as 12 meters) of signalized intersections. Traffic volume are collected at each sensor location together with vehicle *signatures*. A specially designed vehicle re-identification algorithm is developed in [12] to match vehicles from the signatures. The algorithm is based on a statistical model of the signatures, with parameters (such as the threshold to distinguish "matched" and "un-matched" signatures) estimated from the data and no "ground truth" is required. If the algorithm is applied to two consecutive sensor locations (one from upstream and the other one from downstream of an signalized intersection), intersection travel times (or delays) can be obtained directly. A unique feature of vehicle re-identification method is that traffic signal information is not required. In [12], it is further shown that signal phases can also be derived from the matched vehicles by looking at the start and end times of the first vehicle in a queue.

The vehicle re-identification method provides a straightforward way for estimating intersection delays, i.e. by sampled intersection travel times from specially deployed traffic sensors, without the requirement of signal information. Sampled travel times however only provide discrete measurements in the time domain. Now the question is: can we construct a continuous and time-dependent intersection delay pattern using sampled travel times for a given signalized intersection? An intuitive answer is to assume travel times change linearly between two neighboring sampled travel times. As shown in Section 4.2, such a method may not be the most effective especially when the penetration rate is high. In this article, we develop a least-square based algorithm to estimate the delay patterns from sampled travel times by recognizing the underlying characteristics of signalized intersection delays.

The proposed algorithm can be applied to specially deployed fixed-location sensors (such as loop detectors or wireless sensors as shown in [12]) or the Virtual Trip Line (VTL) technique based on GPS-equipped cellular phones as jointly developed and tested by Nokia and the University of California, Berkeley [13, 14]. VTLs are *virtual* loop detectors without any requirement to deploy *physical* detectors or other infrastructures. Instead the concept of VTL leverage on existing infrastructures, i.e. existing GPS-equipped smart phones and cellular networks. In particular, as a vehicle equipped with GPS cell phones passes by a VTL location, the location and speed of the vehicle will be sent to a secure server, from which all vehicles' information will be aggregated and transferred to traffic models. Therefore VTLs can easily be deployed with almost no additional infrastructure cost, which makes it possible to be applied to large-scale arterial networks. Deployment of VTLs is flexible with major considerations for privacy preservation [15]. Arterial VTL data include individual vehicle speeds at each VTL and travel times between consecutive VTLs for those vehicles that are equipped with GPS cellular phones. This type of data provides rich information about arterial traffic state, while privacy violation is maintained at a minimal level.

The raw VTL travel time data can be processed to generate samples of intersection delays. In this article, we propose methods to estimate intersection delay patterns by utilizing these delay samples for single signalized intersections. We first show that intersection delay patterns can be represented as piece-wise linear curves. These curves are developed using well-developed traffic flow theory on queue forming and discharging at signalized intersections. We validate such patterns in microscopic traffic simulation by assuming the knowledge of signal timing and traffic flow information. We then show how to use collected VTL travel times to estimate the parameters of pattern curves, without knowing either traffic signal parameters or traffic volume. In particular, the estimation algorithm is a two-step least square method that can be converted to solve multiple convex quadratic programs in small dimensions. The estimated delay patterns can be directly used to derive signal phases. We test the model and algorithm in microscopic traffic simulation. We also test the estimation algorithm using field experiment data obtained from wireless sensors with reasonable results. We are currently preparing a series of 20-car experiments in the City of Berkeley, CA to further test and verify the proposed intersection delay pattern estimation model and algorithm.

2 Arterial VTL System

For arterials, VTLs are deployed as depicted in figure 1, which is similar to the way wireless traffic sensors are deployed in [12]. In general, we have a VTL placed downstream of each outgoing approach of an intersection. The type of data generated by the VTL system for a pair of VTLs include time crossing the first VTL, travel time between the two VTLs, and the average speed when vehicles crossing each VTL. Speeds are unlikely to be useful because they are highly variable around intersections. Instead, we will use the travel time information to measure delays through the intersection. Note that we get travel time information between intersections in addition to travel times going through the intersection. Given that there will be VTLs deployed to all sides of an intersection, we will be able to get travel time information for any turns of the intersection in addition to going straight through. The fully deployed VTL system will collect updates and push them to a server for processing. The time between pushes will be a consistent interval, typically 1 minute.

3 Approximate Intersection Delay Patterns

3.1 Piecewise Linear Intersection Delay Curves

In this section, we derive models for approximate patterns of intersection delays under normal and over-saturation conditions. The results presented here are based on well-established theories on queue forming and discharging in front of a signalized intersection [16, 17]. The first condition happens when queue can be cleared completely during the green phase of a cycle, while the second condition refers to situations where queue cannot be cleared within one cycle and the residual queue will have to wait for extra time (i.e. more delays) to be cleared. These two condition are the most commonly observed in the field. We notice that under specific situations (e.g. heavy

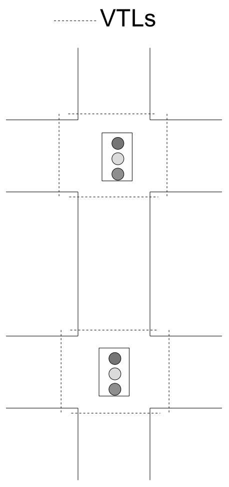


Figure 1: Hypothetical Arterial Road Showing Typical VTL Deployment.

congestion), queues may spillover to upstream intersections and cause further delays. This third condition is however not considered in this article and will be studied in our future research.

Figure 2(a) depicts a typical intersection with two VTLs installed at upstream (VTL1) and downstream (VTL2) respectively. Assume the intersection is signalized. To simplify the discussion, assume that queue never passes beyond VTL1. We can use the bold solid triangles in the figure to represent how queue forms and dissipates in front of the signal (actually those triangles show the waves where two distinct traffic states meet). The horizontal part of the triangles represents the duration of red time. If delays due to vehicle decelerations and accelerations are ignored (which are secondary in terms of intersection delays anyway) and the arrival rate is uniform within one cycle, delays can be fully determined by these triangles. In this figure, dashed lines represent trajectories of vehicles, while dotted lines are boundaries when the discontinuities of delays happen. Note that here we require the vehicle arrival rate is uniform within one cycle (usually less than 2 minutes), which could vary across cycles. If this is not the case, the average arrival rate within the cycle might be used. Further, if the vehicle arrival rate is not uniform but could be approximated by step functions, the left side of the triangle should be represented by piece-wise lines.

We aim to characterize vehicle delay as a function of the time when it passes by VTL1. Note that in reality the measured delay will not be recognized until the vehicle passes VTL2. Here we assume that data have been collected and thus one can do "post-processing" to re-construct a mapping from the time that a vehicle passed VTL1 to its experienced delay at the intersection. Since we assume that the queue never reaches VTL1, as shown by the trajectories of vehicles (dashed lines), if a vehicle approaches the intersection in red time or if the queue length is not zero (e.g. trajectory a in the figure), the vehicle will join the end of the queue first and thus be delayed. The delay encountered by the vehicle is the horizontal part of trajectory a. Otherwise, if a vehicle arrives during green time and there is no queue (e.g. trajectory b), the vehicle will pass the intersection with no delay. As a result, the (red) delay curve at the bottom of Figure 2(a) will spike up at the time that allows a vehicle to travel to the intersection in free flow just before the start of the red time. More importantly, by analyzing the geometry of the triangles, one can observe that if a vehicle passes by VTL1 at a time that would make it get to the intersection just after the start of the red time, delay for this vehicle will be the maximum for the specific cycle. After that, delays will be reduced linearly until no delay is reached. This is represented by the line segments marked as "1" of the delay curve at the bottom of Figure 2(a). The slope of the delay reduction part, denoted as delay reduction rate s, can be calculated analytically as:

$$s = -\frac{u_f(w - u_w)}{w(u_f + u_w)} = \frac{v}{k_j}\left(\frac{1}{u_f} + \frac{1}{w}\right) - 1. \quad (1)$$

Here w is the wave speed, u_f is the free flow speed, u_w is the wave speed when a vehicle joins the queue, k_j is the jam density, and v is traffic flow which is assumed to be constant within a cycle. The three parameters u_f, w, k_j are specific to actual arterial locations, which also determine the fundamental diagram of the location. Notice that since $w \geq u_w$ always hold (refer to the fundamental diagram at the top of Figure 2(a)), s is nonnegative, meaning that the delay always reduces from its maximum (when traffic light turns red) to some minimum value (when light turns green and no queue exists) for normal situations.

To illustrate how equation (1) can be derived, we refer to Figure 2(b). In particular, we assume the delays for a vehicle passing at VTL1 at time t and $t + \Delta t$ are $d(t)$ and $d(t + \Delta t)$ respectively. According to the assumptions made in this paper, the delays at both time instants are corresponding to the lengths of the horizontal lines as shown in the figure. Based on the geometry of the triangles, we have $d(t+\Delta t) - d(t) = \overline{CD} - \overline{A'D'} = \overline{D'D} - \overline{A'C} = \frac{h}{w} - \frac{h}{u_w}$. We also have $\Delta t = \frac{h}{u_f} + \frac{h}{u_w}$, and therefore $h = \frac{\Delta t}{1/u_f + 1/u_w}$. Summarizing the above equations, we have:

$$d(t + \Delta t) - d(t) = \Delta t(\frac{1}{w} - \frac{1}{u_w})(\frac{1}{1/u_w + 1/u_f}). \quad (2)$$

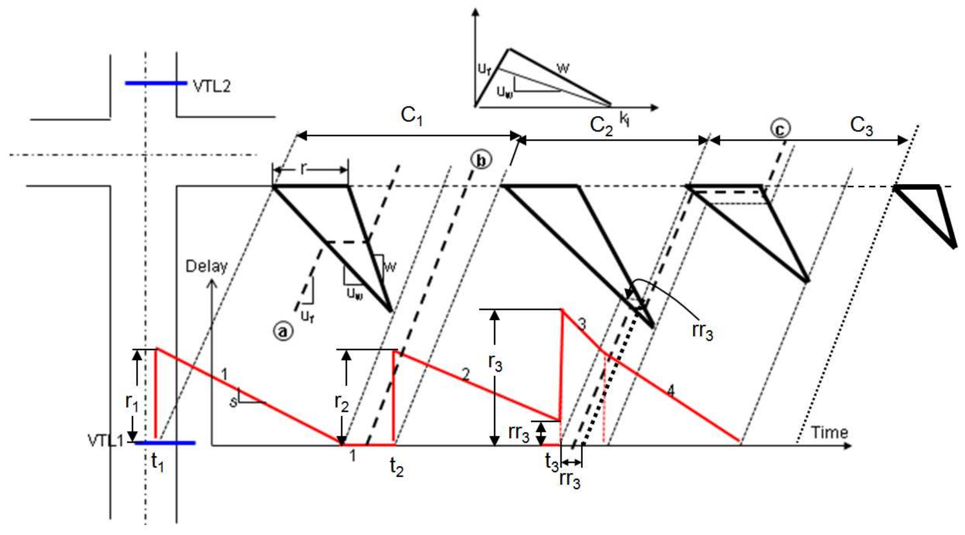
Since the delay reduction rate can be defined as $s = \frac{d(t+\Delta t)-d(t)}{\Delta t}$, we can obtain equation (1) via dividing both sides of (2) by Δt.

Note that the above analysis and equation (1) works only for normal conditions, i.e. no over-saturation or spillover happens. In case of over-saturation, the residual queue from one cycle will have to wait for the next green to be cleared, as shown by trajectory c in Figure 2(a). Under such situation, delay will still be reduced linearly from the maximum value after the start of the red time. However, it will never reach zero; instead, it will have a sudden increase from a nonzero delay to another (local) maximum, indicating the vehicle will have to wait for extra cycle to be cleared. This is marked as "2" in the delay curve in Figure 2(a). After this stage, the delay will be reduced linearly until the impacts of residual queue diminishes, as shown using "3" in the delay curve. The delay will be further reduced in a normal way as marked using "4" in the curve. As a result, the delay curve for over-saturation is still piece-wise linear, but with a more complicated pattern. A distinct feature is that delay is never reduced to zero. The slope of the curves can all be computed analytically by looking at the geometry of the queue forming and discharging triangles.

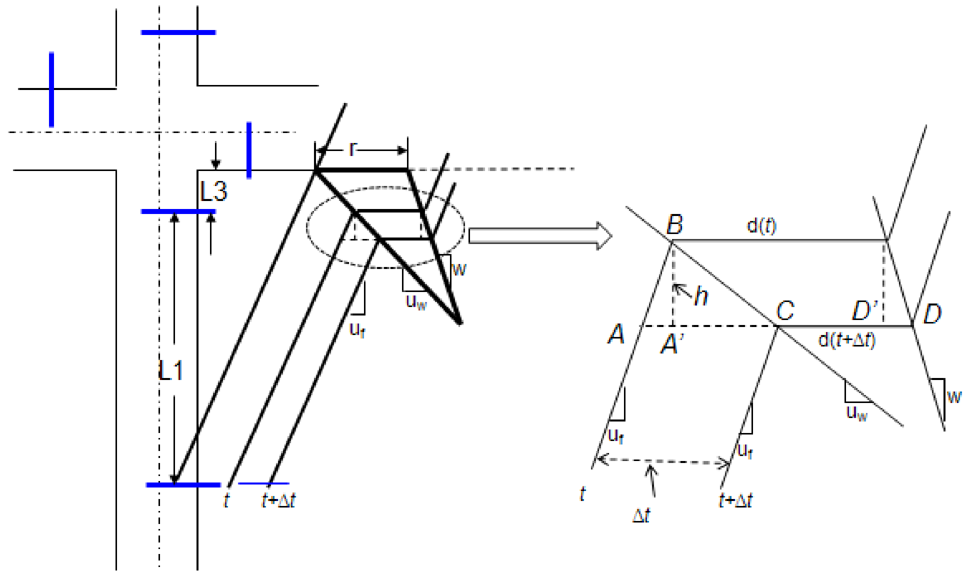
We can see that the approximate delay patterns for signalized intersections (by ignoring the acceleration and deceleration delays) can be represented as piece-wise linear curves. The curves are continuous in most cases, but contain discontinuities (jumps) periodically. These discontinuities correspond to the start of red times, and are important features of intersection delays. In latter sections of this article, we show how this feature can be used to derive signal phases and to estimate delay patterns from measured travel times.

3.2 Validation of PWL Intersection Delay Patterns in Micro-Simulation

In this section, we validate the piecewise linear intersection delay pattern using Paramics, a micro-simulator that is widely used in traffic/transportation studies [18]. Figure 3 depicts a selected signalized intersection in a simulation network in Paramics. We focus on the through movement at this intersection from VTL1 (upstream VTL) to VTL2 (downstream VTL). The distance between



(a) Intersection Delay Patterns



(b) Calculation of Delay Reduction Rate

Figure 2: Theoretical Delay Patterns

these two VTLs is about 1207ft and the free flow travel time is 20.97 seconds under free flow speed 40 mph (miles per hour).

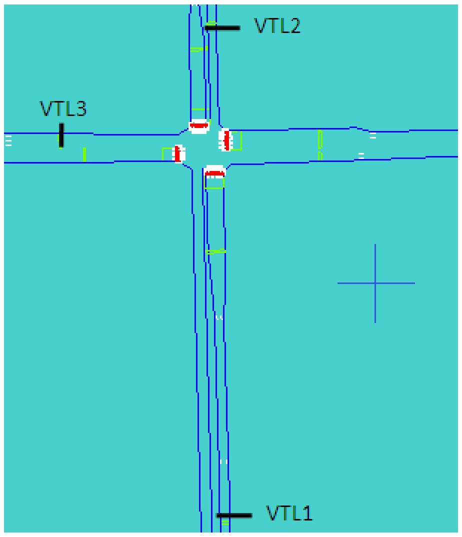


Figure 3: Intersection and VTL Setting Used for the Micro-simulation

We ran the simulation for half an hour from 3:30 to 4:00 pm. The asterisks in Figure 4 depict the simulated travel times for all through vehicles from VTL1 to VTL2. We also collected signal timing parameters from the simulation model, and plotted the red times as (thick) horizontal bars at the bottom of this figure. This way, we know both the signal timing parameters and traffic flow information (since we track individual vehicles' movements in simulation). Furthermore, we observe from the simulation that there was no over-saturation or spillover during the simulation, implying that traffic conditions were normal. Therefore, equation (1) can be used to calculate the delay reduction rate and further to construct the delay pattern curves (see Figure 2(a)). The calculated delay pattern curves are piecewise linear as shown using thin (red) lines in Figure 4. Note also that in Figure 4, instead of displaying delays directly, we show travel times which are the summation of delays and the (constant) free flow travel time.

Figure 4 shows that the approximate piecewise linear delay patterns match very well with the measured travel times (i.e. the asterisks are exactly on or very close to the thin lines, i.e. the delay pattern curves) . This illustrates that the approximate delay patterns are validated using simulation. Determining the exact delay pattern curves, as shown in Figure 2(a), requires the knowledge of signal timing parameters (particularly the starting time and the duration of the red time) and actual traffic volume. This type of data is not available from VTL measurements. The pattern however can be estimated using measured intersection travel times obtained from VTL data. The estimation algorithms are discussed in Section 4.

3.3 Estimation of Signal Phases from PWL Intersection Delay Pattern

The knowledge of the PWL intersection delay pattern enables one to directly estimate signal phases of the intersection. Here we assume a cycle always starts with the red time, implying that the start of the red, the duration of the red, and the end time of the cycle (also the start time of the next red) uniquely determine a cycle. As shown in Figure 2(a), we focus on the Translated Signal Phase Timing (TSPT) at VTL1, which are different from the actual signal phase timing at the intersection by a constant (i.e. the free flow travel time from VTL1 to the intersection). In fact,

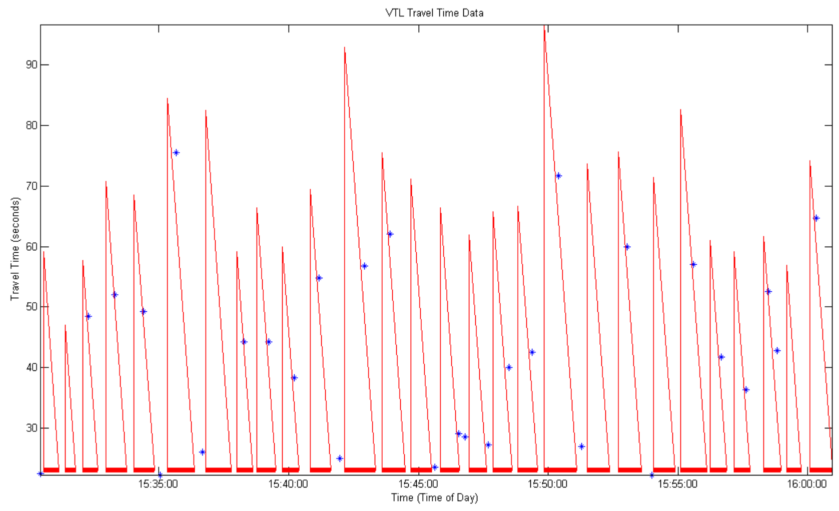


Figure 4: Overlay of Theoretical Delay Curves and Simulated Travel Times

TSPT reflects the times when a vehicle actually "feels" the effect of the signal at VTL1 as if it were just at the intersection. The procedure for estimating TSPT is described as follows.

First, as shown in Figure 2(a), there is a non-trivial increase in delays right after the start of the red time in TSPT (for both normal and over-saturation conditions), with the magnitude of the increase equal to the duration of red time. As delay generally decreases over time within a cycle after the start of red, such increase is a unique feature of intersection delays which only happens at the time when the signal turns red in TSPT. Therefore, detecting such an increase in the measured delays will help identify the start of a new cycle. For example, under normal conditions (e.g. as marked "1" of the delay pattern in Figure 2(a)), the delay increases from 0 to r_1 at t_1 which indicates that t_1 is the start of a cycle (denoted as cycle "C1") in TSPT. This cycle ends when the next increase is detected at time t_2, which also indicates that the next cycle starts at t_2 (cycle "C2"). The duration of the red time is r_1. For over-saturation conditions, the start of red is also associated with such an increase in delay, but needs further adjustment. For example, at t_3 the delay increases from a non-zero value rr_3 to r_3. As illustrated in the figure, the actual start of red (for cycle "C3") in this case is not t_3; rather, it is $t_3 + rr_3$. Similarly, the duration of red is $r_3 - rr_3$ instead of r_3.

In summary, we assume the delay pattern is given, which results in n discontinuities at t_i with delay being increased from rr_i to $r_i, i = 1, \ldots, n$. Then the start of red time is $t_i + rr_i$ and the duration of red is $r_i - rr_i, i = 1, \ldots, n$. This simple procedure is used in later sections to derive signal phase information for both the simulation and field experiment data. Notice that this way we obtain phase information in TSPT, which can be easily translated to actual phase information of the signalized intersection by adding the free flow travel time from VTL1 to the intersection.

4 Estimation Algorithm

The problem investigated in this article is to estimate intersection delay patterns using sampled travel times measured between upstream and downstream locations of a signalized intersection.

The estimation method we propose here is a simple curve fitting algorithm, which is based on two observations regarding the characteristics of intersection delays. First, as shown in Section 3, delay curves are piecewise linear. This enables us to fit the delay measurements using linear forms which significantly reduces the complexity of the fitting algorithm. Second, as shown in Section 3.3, there is a non-trivial increase in delays right after the start of the red time; therefore, detecting such an increase can help identify the start of a new cycle. This turns out to be critical to the estimation algorithm since it breaks the (potentially) large amount of measurements into groups, one for each cycle. The linear fitting within each cycle can then be conducted more straightforwardly.

4.1 A Two-Step Least Square Estimation Algorithm

The estimation algorithm contains cycle breaking and line fitting as two major steps. Figure 5 illustrates how the above two observations can be used in these two steps. We assume there are 16 measured delays, represented by the 16 circles in the figure from a to p. The values of the delays are denoted as $\{d_r, 1 \leq r \leq 16\}$. Each delay is associated with a time stamp, denoted as $\{t_r, 1 \leq r \leq 16\}$. First, by detecting the (non-trivial) increase of delays, we can break the 16 measurements into 4 groups: $\{a, b\}, \{c, d, e, f, g\}, \{h, i, j\}, \{k, l, m, n, o, p\}$. In the figure, s_i, e_i denotes the starting time and ending time of a cycle respectively, which may be defined as the middle point of two consecutive timestamps (one in each cycle) or adjustable based on the calculated average cycle length information (refer to the IDE algorithm in the next subsection). Second, within each cycle, we try to fit the measurements using piecewise linear curves. Figure 5 shows the three typical delay patterns for normal and over-saturation conditions, marked as 1, 2, and 3. Curve 1 is for normal conditions, in which delay reduces linearly until it reaches zero (no delay). Therefore, the delay curve consists of two lines, one with a negative slope and the other one with a zero slope (constant). Curves 2 and 3 are for over-saturation conditions. Curve 2 is a single line with negative slope representing delay reduction (over time) for the first cycle of the overstauration, in which the minimum delay is positive. Curve 3 represents the delay reduction pattern caused by both cycles of the oversaturation, which contains at least 2 lines and may or may not reach zero delay in the end (depending on whether oversaturation disappears in the second cycle).

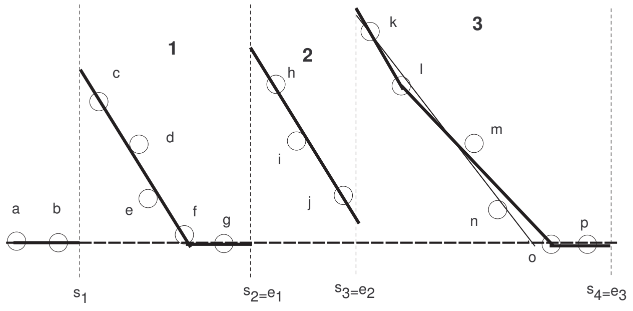


Figure 5: Illustration of the Estimation Algorithm

In summary, although the actual shape of the delay curve within a cycle may vary depending on actual traffic conditions, the fundamental pattern of the curve can be identified as the three cases in Figure 5 for normal and oversaturation conditions. Furthermore, the number of measurements in one cycle tends to be small. For example, consider a three-lane arterial intersection with total traffic volume 1,800 veh/hour. Assume the cycle length is 1 minute, which will result in about 30 (1800/60) measurements under 100% penetration rate. In reality, since the penetration rate is most likely much smaller than 100%, the number of measurements with a cycle will not exceed

1 or 2 dozens. Therefore, although more advanced fitting techniques may be applied, we instead propose a simplistic method based on least square fitting in this article.

The least square method starts with attempting to fit the measurements within one cycle using two straight lines. This is done by enumerating all possible grouping scenarios of the measurements. Denote $\{d_r, r \in R\}$ the set of measurements which is sorted by their timestamps $\{t_r, r \in R\}$. Here $|R|$ denotes the total number of measurements. Then this set of measurements may be divided into two groups by breaking the set at $m = 3, \ldots, |R| - 1$ where m is the starting index of the second group. For each m, fitting can be solved using a convex quadratic program. To see this, we assume the objective of fitting is to reduce the deviation of model predicted and actually measured delays, more specifically the Mean Square Error (MSE) of the predicted delays. We further assume the first line can be represented as $d = a_1 t + b_1$ and the second line can be represented as $d = a_2 t + b_2$. Here a_1, b_1 are parameters for the first line, and a_2, b_2 are parameters for the second line, all need to be estimated. The quadratic problem can then be formulated as follows:

$$\min_{a_1,b_1,a_2,b_2} \sum_{1 \le i \le m-1} (a_1 t_i + b_1 - d_i)^2 + \sum_{m \le i \le |R|} (a_2 t_i + b_2 - d_i)^2 \tag{3}$$

$$\text{s.t.} \quad a_1[(1-\theta)t_{m-1} + \theta t_m] + b_1 = a_2[(1-\theta)t_{m-1} + \theta t_m] + b_2. \tag{4}$$

In the above model, the objective in (3) is the summation of MSE of the two groups: the first group contains data points $1, \ldots, m-1$ and the second group contains data points $m, \ldots, |R|$. Notice that we assume m is given, and $a_1 t_i + b_1$ is the predicted delay at t_i using the first line whose actual delay is d_i for any $1 \le i \le m-1$. Similarly, $a_2 t_i + b_2$ is for the delay predicted by the second line. The constraint (4) is required because the two lines have to intersect at the boundary of the two groups. Here we assume the boundary is at $0 \le \theta \le 1$ from t_{m-1} with respect to the difference between t_m and t_{m-1}, by noticing that $m-1$ is the last measurement in the first group and m is the first measurement in the second group. As a special case, if the boundary is at the middle point of t_m and t_{m-1}, we have it at $(t_{m-1} + t_m)/2$. Clearly the above model has only 4 variables and is convex and quadratic, which can be solved very efficient using standard quadratic program solvers.

The quadratic model (3) - (4) will be solved for any $3 \le m \le |R| - 1$, resulting in $|R| - 3$ solves. The minimum objective value of all solves is denoted as f_2. We compare f_2 with the objective value of fitting all measurements using one line, denoted as f_1. If $f_2 < f_1$, the 2-line fitting is accepted; otherwise, the one line fitting is accepted. If 2-line fitting is accepted, the algorithm will further test if the duration of either group is larger than a threshold. If yes, the above process will be repeated on the group trying to fit the group with two new lines. This process repeats itself until either all groups are represented as a single line or the duration of the group is below the threshold. As one example, the third case in Figure 5 can be first represented by two lines via solving the quadratic model: the first one with a negative slope shown as the thin line and the second one with zero slope (constant). The first line can be further represented by the two bold lines by solving the quadratic model again.

We summarize the estimation algorithm as follows, which is denoted as IDE (short for Intersection Delay Estimation) algorithm.

IDE Algorithm

Step 1. Initialization. Collect VTL travel time data and process them to obtain intersection delays. Set two thresholds, th_1 and th_2.

Step 2. Cycle Breaking. Scan all the delay measurements and detect if the delay increase from one measurement to the next one exceeds th_1. If yes, break the cycle at the second measurement. This step will produce groups of delay measurements.

Step 3. Curve Fitting within A Cycle. Denote $\{d_r\}, \{t_r\}, \forall r \in R$ all the measurements in a given cycle.

3.1 Solve the convex quadratic program (3) - (4) for all $3 \leq m \leq |R| - 1$. Here $\theta = 0.5$ is used, i.e. the boundary is at the middle point. Denote the minimum objective value among all $|R| - 3$ solves is f_2.

3.2 Solve the least square fitting problem using a single line and denote its objective value as f_1.

3.3 If $f_2 > f_1$, fit the delay pattern using the single line. Otherwise, represent the delay curve using two lines. If the duration of either line is larger than th_2, set $\{d_r\}, \{t_r\}, \forall r \in R$ as the measurements corresponding to this line and go to Step 3.1. Go to Step 4 if Step 3 is done for all cycles.

Step 4. Cycle Length Adjustment (for pre-timed or actuated coordinated signals). Calculate the average cycle length via dividing the total time period by the number of cycles detected. Using this average cycle length, adjust the boundaries of each cycle, i.e. θ, so that each cycle length is as close as possible to the obtained average cycle length.

Step 5. Stop with the optimized delay pattern curves.

In the IDE algorithm, th_1 is the threshold for the increase of delay to detect the start of a new cycle, while th_2 is the threshold of the time window to break measurements within a group into possibly more cycles. The value of th_1 should be exactly the duration of the red time in ideal situations: vehicles passing by in green time experience exactly the free flow travel time and the delay of the vehicle that arrived just before the start of the red time is also captured. In reality, due to travel time variations across individual vehicles and most importantly the fact that only samples are available, one can set $th_1 = \alpha_1 R$, where R is the duration of the red time and α_1 is an coefficient. Similarly, th_2 can be selected as the cycle length in ideal cases. In practice, we can set $th_2 = \alpha_2 C$, where C is the cycle length and α_2 is another coefficient. The selections of α_1 and α_2 may be location specific and need further investigations. In this paper, we set th_1 and th_2 as 15 seconds and 35 seconds respectively, which seems work fine for the simulation data and field observations. Step 4 is a fine-tuning step for pre-time or actuated coordinated signals. As for these types of signals, the cycle length is usually fixed. The average cycle length via the first three steps can hopefully provide an indication of what this fixed cycle length might be. This information can then be used to adjust boundaries, i.e. θ, of each cycle so that the cycle length is close to the average cycle length. Notice that Step 4 should not be conducted for isolated actuated signals as the cycle length may vary for this type of signals.

It is easy to see that the most computationally expensive step is Step 3 particularly Step 3.1 for solving the multiple convex quadratic programs. Assume the maximum number of measurements in one cycle is N, which as mentioned before should be no more than 30 in most cases. Then we immediately have $|R| \leq N$. As shown in [19], convex quadratic programs can be solved in polynomial time, particularly $O(n^3 L)$ with n the number of variables (4 in our model) and L is the size of the problem encoding in binary as defined in [19]. Therefore, the time required in Step 3.1 is $(|R| - 3)O(n^3 L)$, which is no more than $O(n^3 LN)$. Step 3.1 may need to be executed for multiple times, which should not exceed $|R|$. As a result, the maximum possible complexity for Step 3.1 is less than $O(n^3 LN^2)$. Denote further the maximum number of cycles that can be obtained in Step

2 is K. Clearly, the overall complexity of the IDE algorithm is $O(KLn^3N^2)$ for the worst case. Therefore, the IDE algorithm proposed in this article is polynomial and can be implemented in real time applications.

The above discussions also show that in order to appropriately estimate the delay curves, we need at least two measurements per cycle for normal conditions. For oversaturation-conditions, this number will be at least 4. If the cycle length is 1 minute, the required minimum sample rate is 120 vph (vehicle per hour) for normal conditions and 240 vph for over-saturation conditions. If we consider a two-lane arterial street with traffic volume 1200 vph, this implies a minimum penetration rate of 10% for normal conditions and 20% for over-saturation conditions. Since this penetration rate has to hold for each cycle, in practice we expect that the (average) penetrate rate should be 30% - 40% or higher in order to provide 20% penetration rate for each cycle in most cases.

It is worth noting that the IDE algorithm only uses measured travel times as input, without assuming knowledge of signal timing parameters or traffic volume information. This is a fundamental difference between IDE and previous models for estimating intersection delays or travel times.

4.2 Test of the Algorithm in Micro-Simulation

In this subsection, we assess the performance of the IDE algorithm using the Paramics model illustrated in Figure 3. We focus on the left turn movement at this intersection from VTL1 to VTL3. The distance between these two VTLs is about 1273ft and the free flow travel time is 21.69 seconds under free flow speed 40 mph. To account for the fact that vehicles need to slow down to make the left turn, we added 5 seconds to the free flow travel time, making it to 26.69 seconds.

We ran the simulation for 1 hour from 3:30 to 4:30 pm. Figure 6(a) first depicts the simulated travel times from VTL1 to VTL3 for all vehicles making left turns. The travel times look purely random at the first glance. For comparison purposes, we display the duration of red times for this left turn as the horizontal bars at the bottom of the figure. Note that these durations are "ground-truth" and obtained directly from the simulation model. We then applied the IDE algorithm on the simulated travel times and the identified delay patterns are shown in Figure 6(b). Note that in this figure, the curves are actually for travel time patterns, which is exactly the same as the delay patterns (with a constant difference, i.e. the free flow travel time). We can see that the estimated patterns match very well with the measured travel times (represented as asterisks in the figure). The estimation errors, i.e. $\hat{d}_i - d_i$, are shown using plus signs in the figure. Here $\hat{d}_i$ is the estimated delays. It is easy to see that most estimation errors are close to zero, implying that the estimation quality is high. To further quantify the estimation quality, we define a quality measure that is the percentage of estimates with errors no more than 15% of the measured travel times. Denote this quality measure as α, which can be defined as follows:

$$\alpha = \text{Prob}\left(|\frac{\hat{d}_i - d_i}{d_i + fftt}| \leq 0.15 \right). \tag{5}$$

Here $fftt$ denotes the free flow travel time from VTL1 to VTL3. Note we add $fftt$ to the denominator of the right side of (5) since d_i may be zero. In this sense, equation (5) is actually the error defined for travel times. Clearly, the estimation quality becomes higher for larger α. For this particular example, we get $\alpha = 99.32\%$, which indicates that the IDE algorithm works very well for estimating delay or travel times patterns.

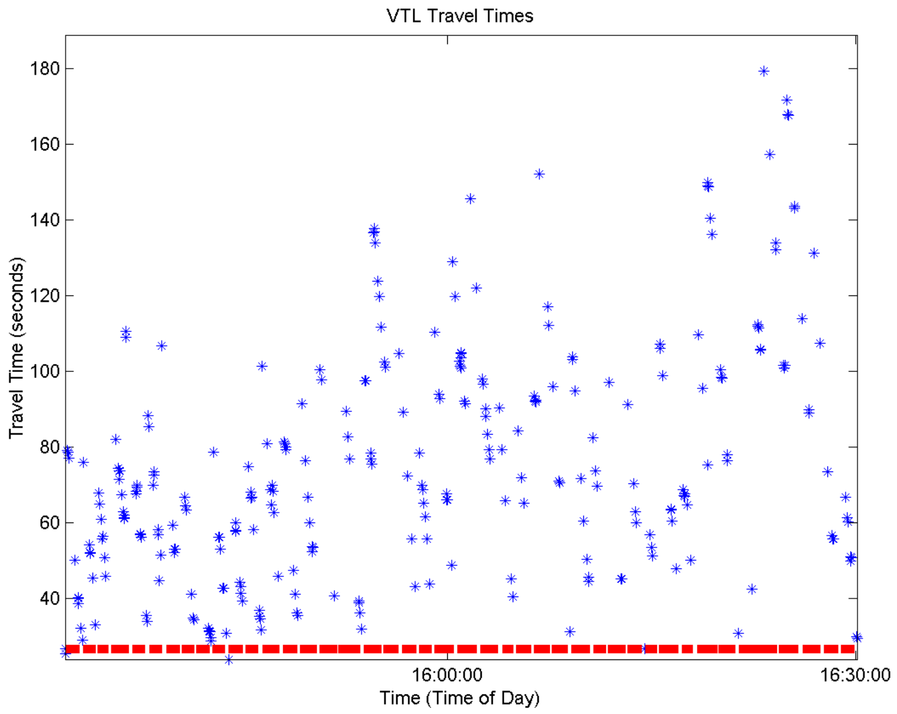
Notice that during this 1 hour simulation, we have both normal conditions and over-saturation conditions, as indicated in Figure 6(b). These conditions are verified in the actual simulations. Also, by comparing the delay patterns with the "ground truth" red times on the bottom of the figure, we can further verify the patterns are associated with signal timing properly. Again, the red times are shown here only for illustration purposes, and are not used in the IDE algorithm.

One of the major reasons for the good performance of the IDE algorithm in the above example is due to the fact that all vehicle travel times are assumed to be known. In other words, the penetration rate is 100%. The next question to ask is how the penetration rate will influence the estimation quality. To answer this question, we random sample, for a given penetration rate p, the measured travel times; we then use the sampled travel times to estimate the delay patterns. For this purpose, we assume the probability of selecting a particular measurement is p. Such sampling results in two sets: the first set contains travel times that were selected, and the second set consists of all un-selected travel times. We then utilize the first set to estimate delay patterns by the IDE algorithm, while the second set is used for testing the estimation quality.

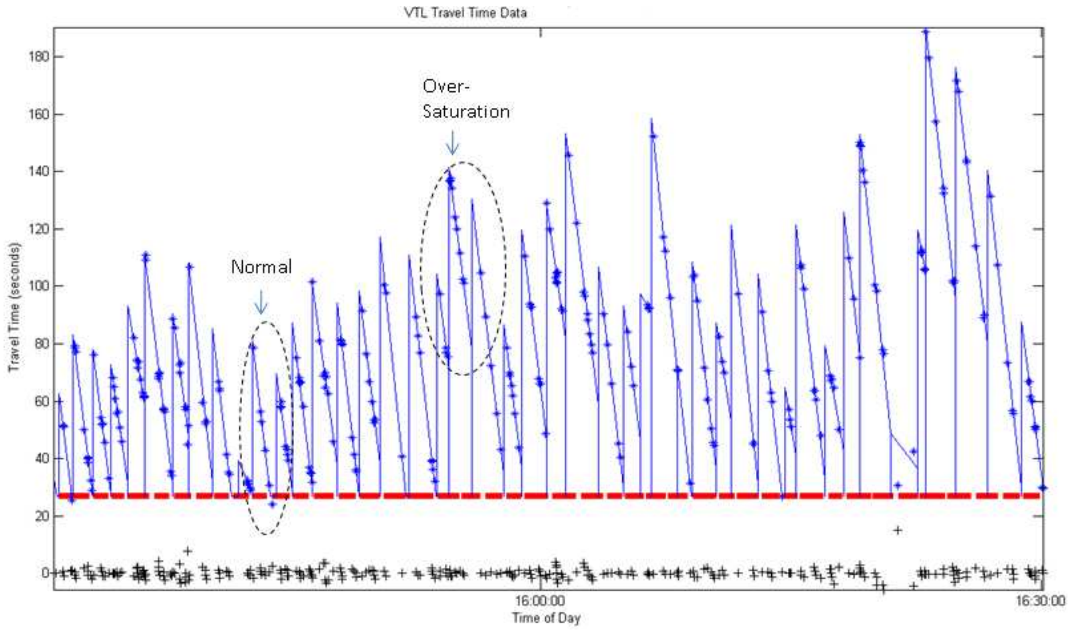
Figure 7 depicts the impacts of penetration rates on the estimation quality. In the figure, we vary penetration rate from 6% to 100% using 2% as the increment. For each penetration rate, we ran the random sampling procedure for 50 times. Each time, the sampled travel times were used to estimate delay patterns and the unselected travel times were used to test the estimation quality, i.e. to compute α. The plus signs in Figure 7 represent the α's and the solid line is the average of the 50 runs, both for the IDE algorithm. For comparison purposes, we also calculated the estimation by pure linear interpolation. That is, for each sampling run, the sampled travel times are treated as grid. The unselected travel times can then be estimated by assuming travel times change linearly between any two adjacent travel times. This linear interpolation represents an naive approach to provide travel time (delay) estimation based on sampled travel times. In Figure 7, dots represent α's for each sampling run under a given penetration rate, and the dash line is the average across all 50 runs, both for the linear interpolation approach.

We can see from this figure that at least for this particular example, if the penetration rate is less than 25interpolation approach is superior to the IDE algorithm. However, as the penetration rate increases, the IDE algorithm becomes more effective in estimating delay patterns. If the penetration rate exceeds 40%, this difference is larger than 10%, indicating that the IDE algorithm is significantly better than the linear interpolation approach. Such a trend remains pretty constant as the penetration rate further increases. This observation is consistent with the discussion for penetration rate following the description of the IDE algorithm in Section 3.

Using the procedure outlined in Section 3.3, we also estimated the timing of the intersection signal phases. This was conducted using different penetration rates ranging from 25% to 100%. Figure 8 depicts the estimated signal phases with the solid horizonal bars representing the duration of red times. On the top of the figure, the "ground-truth" signal phases from the simulation are also shown for comparison purposes. The solid vertical lines illustrate the start of red time (also the start of a cycle) from the ground-truth signal phases, and the vertical dashed lines indicate the end of red times. From the figure, we can observe that at high penetration rates ($> 60\%$), the estimated phases are close to the true phases, both in terms of duration of cycles (or red times) and the actual timing. The results however deteriorate quickly as the penetration rate becomes smaller.



(a) Travel Times for Simulation Data



(b) Intersection Delay Patterns for Simulation Data

Figure 6: Micro-Simulation Results: Travel Times and Delay Pattern Estimates

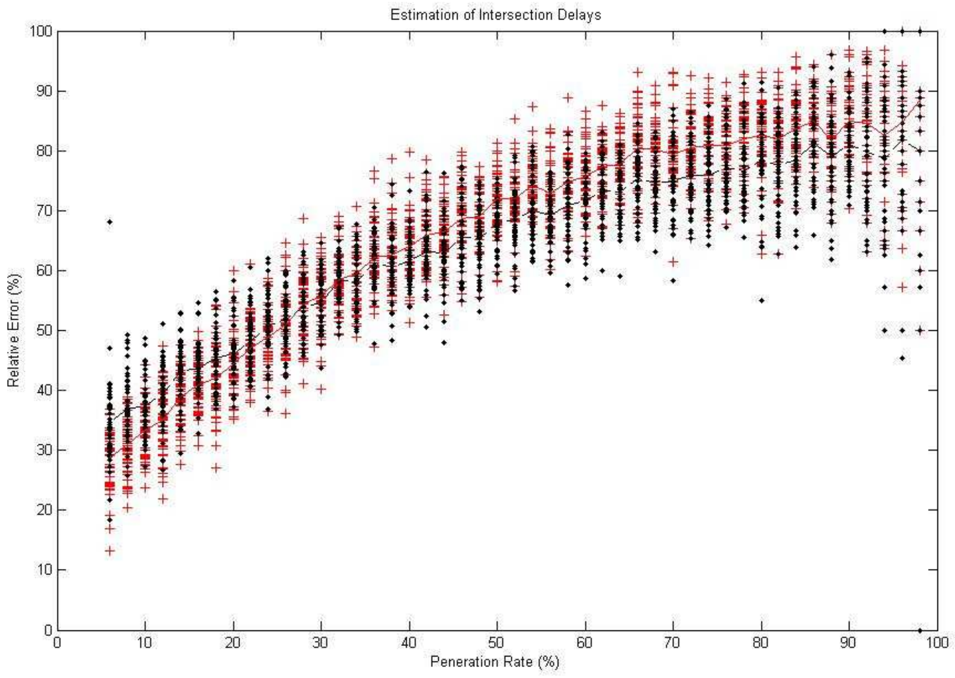


Figure 7: Impact of Penetration Rate on Delay Pattern Estimation Quality

5 Results of Field Experiment

We test the PWL intersection delay model and the estimation algorithm using data from a field experiment. The test site is the intersection of San Pablo Ave and Solano Ave in Albany, CA, as shown in Figure 9(a). The data were obtained from two sets of wireless traffic sensors, installed upstream and downstream of the subject intersection respectively. The raw data collected from those sensors contain traffic flow and vehicle signatures. A re-identification algorithm was applied to match vehicles. Travel times between the two sets of detectors can then be obtained from the matched vehicles. For detailed descriptions of the test site and the vehicle re-identification algorithm, one can refer to [12]. In this article, we use the travel times from matched vehicles directly. In particular, the data contains travel times of 140 vehicles for a 30-minute period (1:00 - 1:30 pm). The travel time data are shown as asterisks in Figure 9(b).

We applied the IDE algorithm in Section 4.1 to the travel times in Figure 9(b). The estimated delay pattern curve is shown as thin solid lines in Figure 9(b). The plus signs represent estimation errors (between asterisks and the delay pattern). It turns out that for this data set, nearly 88% of vehicles will have an estimation error less than 15% if the estimated delay pattern is used. This illustrates that the delay pattern is a fairly good estimation to the ground truth travel times. In the figure, we further highlighted three cycles, C_1, C_2, and C_3, during which over-saturation happened. In fact, over-saturation happened during both C_1 and C_2 as marked in the figure. During C_3 all queues were cleared up. As a result, the delay pattern for C_3 consists of three line segments, as indicated in the figure. In particular, the first line segment represents delays caused by all three cycles, the second line segment represents delays caused by C_2 and C_3, and the third line is for delays caused by C_3 only.

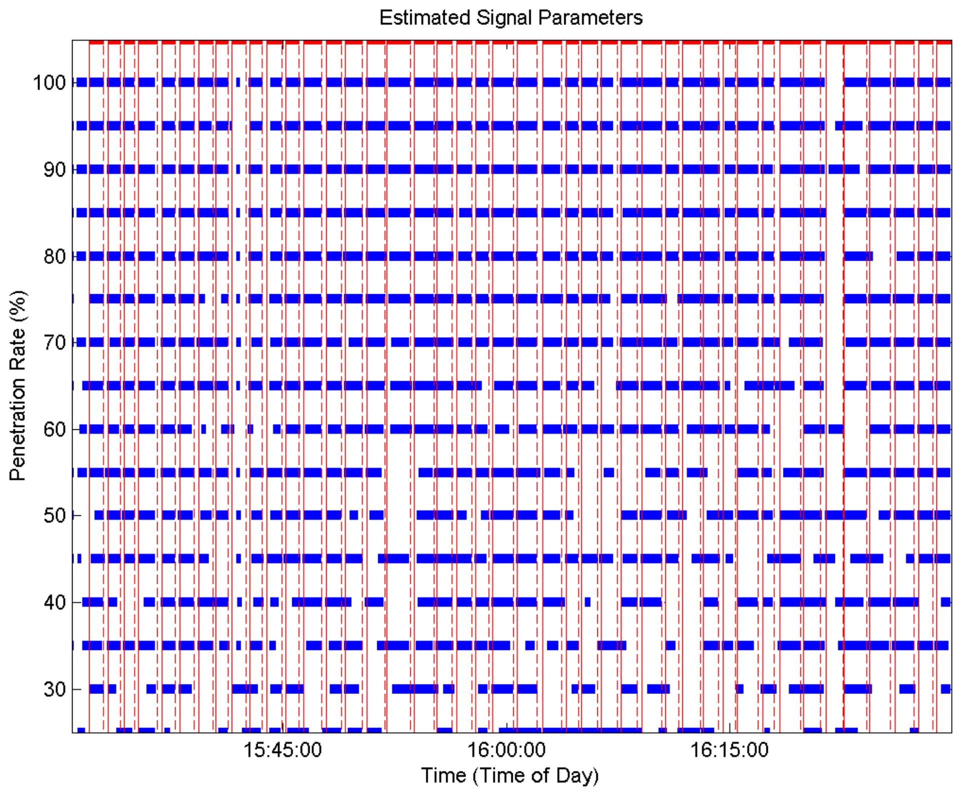
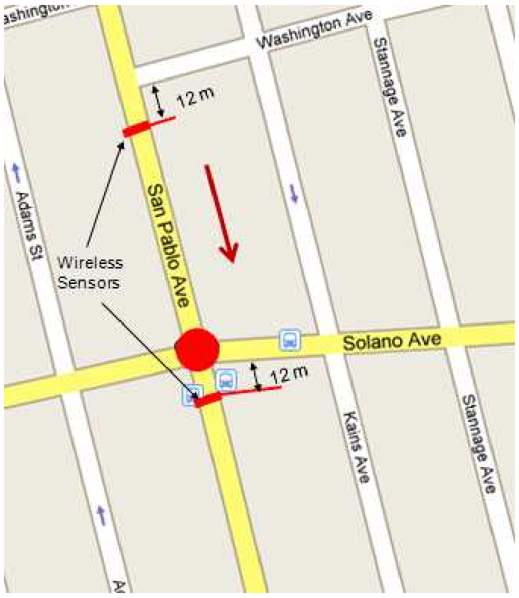


Figure 8: Estimated Signal Phases for Simulation Data

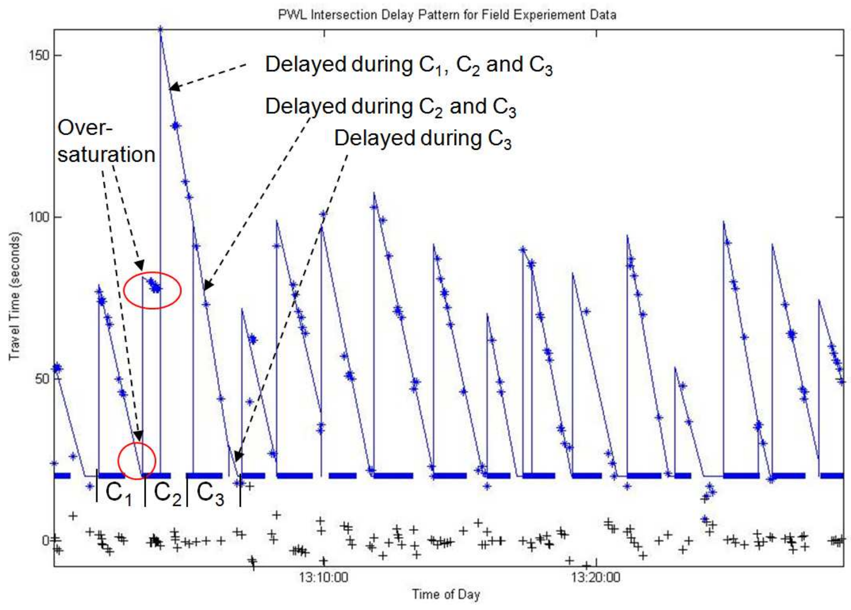
We also applied the signal phase estimation procedure in Section 3.3 to this data set. As this intersection is actuated coordinated with cycle length 108 seconds, Step 4 of the IDE algorithm is applied. The average cycle length generated by the IDE algorithm is 106 seconds, which is very close to the true cycle length (the difference is less than 2%), which indicates that at least the cycle breaking algorithm works fine. The estimated phases (red times) are shown in Figure 9(b) using thick horizontal bars. Table 1 lists the cycle lengths and the durations of red times, as well as the deviation (in percent) between the estimated cycle lengths and the true cycle length. From the table, most of the estimated cycle lengths (13 out of 16, or 81%) are within 15% of the true cycle length. The worst case is an underestimate of nearly 27%, while the best case is an underestimate of only 0.7%. This shows that the IDE algorithm proposed in this paper works fairly well for the field experiment data. The reason the estimation algorithms work fine is mainly due to the fact that the vehicle re-identification algorithm can match 45-65% of total vehicles [12], i.e. the penetration rate of the data set is 45-65%. As discussed in Figure 4.2, such high penetration rate will generate reasonable estimation of delay patterns and signal phases.

6 Conclusions and Future Plan

We proposed in this article a two-step algorithm to estimate arterial signalized intersection delay patterns under both normal and oversaturation conditions. First by investigating the queue forming and discharging process at signalized intersections, we showed that intersection delays can be



(a) Test Site of Field Experiment (Source: maps.google.com)



(b) Estimated Delay Pattern and Signal Phases for Field Experiment Data

Figure 9: Field Experiment: Test Site and Delay Pattern Estimation

Index	Cycle Length (sec)	Red Time (sec)	True Cycle Length (sec)	Deviation (%)
1	94	31	108	-13.4
2	98	59	108	-9.4
3	96	61	108	-11.5
4	123	79	108	13.5
5	81	52	108	**-25.3**
6	114	75	108	5.9
7	97	59	108	-10.2
8	134	88	108	**24.0**
9	117	71	108	8.4
10	79	50	108	**-26.9**
11	111	70	108	2.4
12	120	63	108	11.5
13	106	74	108	-1.5
14	107	34	108	-0.9
15	107	79	108	-0.9
16	107	72	108	-0.7

Table 1: Estimated Signal Phase Parameters

represented as piecewise linear curves. In particular, after the start of the red time, there is always a significant increase in the delay pattern. This unique feature helps detect the start of a cycle, which in turn makes it possible to break potentially large data samples (i.e. measured travel times) into groups roughly equivalent to signal cycles. We then developed a least square based linear fitting algorithm to estimate the delay pattern within a cycle. We showed that the least square method can be converted to solve multiple convex and quadratic programs each with only four variables. Therefore the proposed delay pattern estimation algorithm is polynomial in time and can be implemented in real time applications. We further developed a procedure to derive signal phases based on estimated delay patterns. We tested the model and algorithm using microscopic traffic simulation data and field experiment data. The results illustrated that the IDE algorithm is promising when the penetration rate is relatively high (e.g. larger than 40%).

The proposed model and algorithm only requires sampled travel times obtained between consecutive locations in arterial streets, most likely upstream and downstream of an intersections. This is in contrast to most previous intersection delay or travel time models that assume at least signal timing parameters and traffic flow information. As a result, the intersection delay model and algorithm have the potential to be applied in large scale arterial networks, especially if integrated with the VTL technique designed for GPS-equipped cellular phones.

The presented work in this article is only the first step in developing arterial delay models. Some future research directions can be summarized as follows:

1. We only considered normal and oversaturation conditions in this article. The authors are now working on characterizing delay patterns under spillover conditions and results will be reported separately.

2. The least square based IDE algorithm only considers the two most significant features of intersection delays, and is currently working well for relatively high penetration rates (e.g. $> 40\%$. The algorithm needs to be refined by exploring more the characteristics of arterial traffic flow, traffic signal systems, and delay patterns. For example, if the vehicle arrival rate is not uniform within one cycle, the average arrival rate might be used. However how this will impact the performance of the model needs further investigations.

3. We tested the model and algorithm using micro-simulation and data from a field experiment. A series of field experiment is currently underway to deploy 20 cars in the City of Berkeley, CA to collect arterial travel times, which will be used to test the proposed model. Results will be reported in subsequent articles.

Acknowledgement

The authors appreciate Dr. Pravin Varaiya at the University of California, Berkeley for providing the field experiment travel time data. The first author also thanks Dr. Varaiya and Dr. Andy Chow for several insightful discussions on arterial modeling.

References

[1] H.E. Gault and I.G. Taylor. The use of output from vehicle detectors to access delay in computer-controlled area traffic control systems. Technical Report Research Report No. 31, Transportation Operation Research Group, University of Newcastle upon Tyne, 1977.

[2] V.P. Sisiopiku and N.M. Rouphail. Travel time estimation from loop detector data for advanced traveler information system applications. Technical report, Illinois University Transportation Research Consortium, 1994.

[3] H.M. Zhang. A link journey speed model for arterial traffic. *Transportation Research Record*, 1676:109–115, 1998.

[4] X. Xie, R.L. Cheu, and D.H. Lee. Calibration-free arterial link speed estimation model using loop data. *Journal of Transportation Engineering*, 127(6):507–514, 2001.

[5] A. Skabardonis and R. Dowling. Improved speed-flow relationship for planning applications. *Transportation Research Record: Journal of the Transportation Research Board*, 1572:18–23, 1997.

[6] H. Xiong and G. Davis. Travel time estimation on arterials. In *Proceedings of the 87th Annual Meetings of Transportation Research Board (CD-ROM)*, 2008.

[7] A. Skabardonis and N. Geroliminis. Real-time estimation of travel times on signalized arterials. In *16th International Symposium on Transportation and Traffic Theory*, pages 387–406. 2005.

[8] H. Liu and W. Ma. A virtual probe approach for time-dependent arterial travel time estimation. *Presented at the 87th Annual Conference on Transportation Research Board, and Submitted for publication*, 2008.

[9] C. Oh. *Anonymous Vehicle Tracking for Real-Time Traffic Performance Measures.* PhD thesis, University of California, Irvine, 2003.

[10] S.G. Ritchie, S. Park, S.T. Jeng, and A. Tok. Anonymous vehicle tracking for real-time freeway and arterial street performance measurement. Technical Report Research Report, UCB-ITS-PRR-2005-9, California PATH.

[11] C. Oh and S.G. Ritchie. Anonymous vehicle tracking for real-time traffic surveillance and performance on signalized arterials. In *Proceedings of the 82nd Annual Meetings of the Transportation Research Board (CD-ROM)*, 2003.

[12] K. Kwong, R. Kavaler, R. Rajagopal, and P. Varaiya. Arterial travel time estimation based on vehicle re-identification using wireless sensors. *Submitted to Transportation Research, Part C*, 2008.

[13] S. Amin et al. Mobile century-using gps mobile phones as traffic sensors: a field experiment. In *Proceedings of the 15th World congress on ITS*, 2008.

[14] D. Work, O.P. Tossavainen, S. Blandin, A. Bayen, T. Iwuchukwu, and K. Tracton. An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices. In *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.

[15] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, JC Herrera, and A. Bayen. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)*, Breckenridge, U.S.A., June 2008.

[16] G.F. Newell. *Theory of Highway Traffic Signals*. Institute of Transportation Studies, University of California, Berkeley, CA, 1988.

[17] N.M. Rouphail, A. Tarko, and C. Li. Traffic flow at signalized intersections. In *Traffic Flow Monograph*, pages 9.1–9.32. 2008.

[18] L. Chu, X. Liu, and W. Recker. Using microscopic simulation to evaluate potential intelligent transportation system strategies under nonrecurrent congestion. *Transportation Research Record*, 1886:76–84, 2004.

[19] D. Goldfarb and S. Liu. An $o(n^3l)$ primal interior point algorithm for convex quadratic programming. *Mathematical Programming*, 49:325–340, 1990.