

Self Learning Tool for Travel Time Estimation in Signalized Urban Networks Based on Probe Data

Charitha Dias Marc Miska Masao Kuwahara

University of Tokyo, Institute of Industrial Science, 4-6-1 Komaba, Meguro-ku, Tokyo, Japan 153-8505

Abstract

In this paper we are introducing a self learning tool for travel time estimation in signalized urban networks based on probe data. The main feature of this tool is, that it can be applied with a basic network description instead of a detailed modeling of the network structure. We show how probe data can be utilized to train a Bayesian network to forecast the travel time on a route along an arterial road and how the forecast performs under different initial conditions. In the conclusion we take a critical look on the limitations of such a system and possible extensions to increase its performance.

Keyword:

travel time estimation, probe data, data mining

Introduction

This paper deals with travel time estimation in signalized urban networks. The approach utilizes probe data to collect delay times at intersections and the probabilities of stopping at them. This information is used to train a Bayesian network, which learns from the given patterns and is able to compute possible travel times along the route with their probabilities. We will show the systems performance under several initial conditions and discuss its behavior. As an additional feature we show that the developed system needs just basic network information instead of a detailed configuration of signal timings and infrastructure. In the conclusion we take a critical look at the system and point out limitations of such a system and possible extensions in future work.

Methodology

When vehicles are approaching an intersection, they either have to stop at the intersection or can pass without delay. These “Stop and Go” patterns as well as the associated delays depend on the upstream demand, weather conditions, incidents etc. When a vehicle is traversing along the route, possible events for the route can be defined based on the conditional events experienced at each intersection. Those possible events for a route can be described in a binary tree structure” as shown in Figure 1.

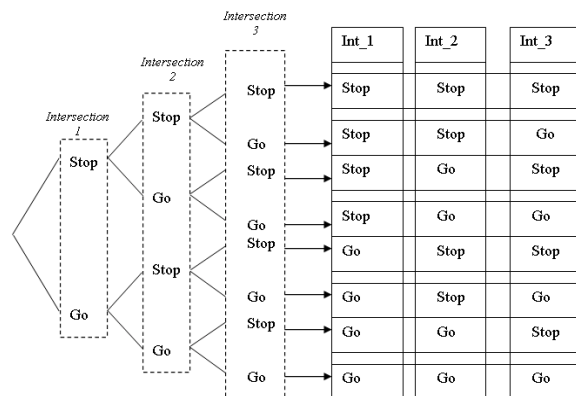


Figure 1 Binary tree structure of conditional events traversing a route with 3 intersections

Although an infinite number of travel times are possible along a road section, based on Figure 1 we can categorize those into categories (here 2^3). The events at intersections (“stop” or “go”) can be described with their probabilities and can be represented as a vector we define as *Probability Vector*.

When stopping, vehicles experience a delay. Therefore, the above mentioned events are associated with delay values. These delay values for all possible events along a given route can also describe by a vector we call *Delay Vector*.

Based on those probability and delay vectors we developed a self-learning tool to estimate travel times on signalized arterial roads with their probabilities. Figure 2 shows an overview of the system.

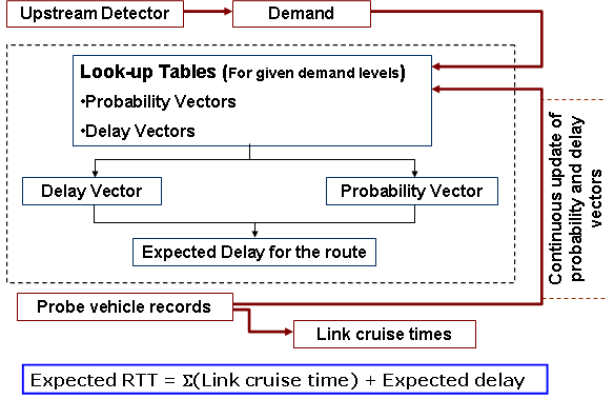


Figure 2 Self Learning Travel Time Estimator

The system consists of probability vectors and delay vectors for different demand levels. The relevant delay vector and probability vector are selected based on the upstream demand level, observed by counting detectors. Based on these two vectors the expected delay for the route is estimated as:

$$E(D) = \bar{P} \cdot \bar{D}^t$$

With

$E(D)$ – Expected Delay

$\bar{P} = \{p_1, p_2, \dots, p_j, \dots, p_J\}$

$\bar{D} = \{d_1, d_2, \dots, d_j, \dots, d_J\}$

p_j – Probability of event j occurrence, $1 < j < J$

d_j – Delay associated with event j , $1 < j < J$

Link cruise times can be obtained from probe records directly. With those link cruise times and the calculated expected delay value, the expected route travel time for the route is calculated as follows:

$$E(RTT) = \sum_{i=1}^n LCT_i + E(D)$$

with

$E(RTT)$ – Expected Route Travel Time

LCT_i – Link Cruise Time of Link i

n – Number of Links along the Considered Route

$E(D)$ – Expected Delay

The upstream demand level, incidents and weather etc. have their effects on conditional binary events and delay values at intersections as well as on link cruise times. These relations were modeled in a Bayesian Structure as

shown in Figure 3.

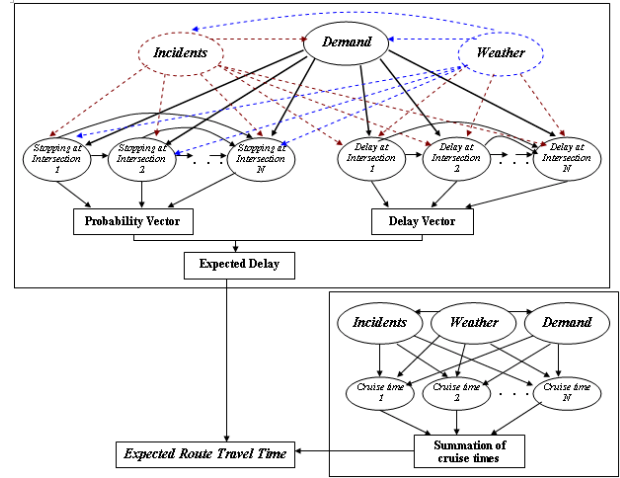


Figure 3 The Bayesian Structure for Route Travel Time Estimation

The circles are measurements (*evidences*) where as rectangles are the belief we obtain using those measurements. Basically there are several evidences:

- Demand (Measured with detectors)
- Weather (Monitored with suitable sensors)
- Incidents (Monitored with video cameras etc.)
- Stopping or not stopping at each intersection
- Delay experienced at each intersection (if stopped)
- Free flow link travel times

We can identify two basic components in Figure 3. First, the component for calculating expected delay and second, the component for calculating cruise times. Further, in case of the expected delay component, there are two sub components to determine the delay and probability vectors.

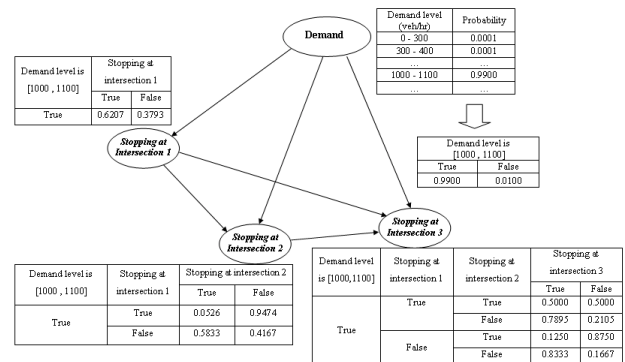


Figure 4 Conditional Probability Tables to Estimate Probability Vector

Figure 4 gives a more detailed view on the component to determine delay vector using sample conditional

probability tables. Since in this study we neglected the effect of incidents and weather on delay vector and probability vector, we have not included incidents and weather.

Since the proposed system is a learning system that depends on probe vehicle information, in this study, we investigate three kinds of initialization for the probability and delay vectors and their effect on the learning performance. These are:

'Initially estimated' vectors

In this case an initial approximation is used for probabilities of stopping or not stopping at intersections based on cumulative arrival and departure curves for the route. First cumulative arrival departure curves for the route are constructed for given demand levels considering basic network information such as geometry (link lengths), signal settings (cycle times, phase times, offsets etc.), speed limits and acceleration / deceleration effects. Based on that, an initial approximation is given for probability and delay vectors.

"50% - 50%" vectors

Here we assume that there is an equal chance to be delayed or not delayed at each intersection. Therefore, the resulting probability vector gives equal probabilities for all possible events. In case of delays we assign the maximum possible delay for a vehicle (i.e. the red phase time) that stops at the intersection. Accordingly, the initial delay vector is constructed for all possible events.

'Initially Empty Vectors'

In this case probability and delay vectors do not have any value initially, which means that initially the expected delay for the route is zero. Then probability and delay vectors are created with the first set of probe vehicles and updated with the probe vehicles arriving in the next updating step. The special feature of this case is that the probability and delay vectors are fully created by evidence from probe vehicles.

Once we have initialized our system, it is ready for operations. Then the system is updated during operation, with the available probe vehicle information.

Initial probability vectors are updated after a fixed time period. That is, evidences from all probe vehicles are collected within a fixed time period (say 5 minutes) and the initial probability vector for the relevant demand level is updated. To update the probability vector software named UnBBayes developed by Fernandes (2004) was used.

UnBBayes uses *K2 Algorithm* which heuristically searches for the most probable belief-network structure given a database of cases to estimate conditional probabilities. Basically, *K2 Algorithm* calculates the probability of the database *D*. (Ruiz, 2005). The initial delay vector is updated as follows:

$$D_{upd,n} = \left(\frac{n-1}{n} \right) D_{n-1} + \left(\frac{1}{n} \right) D_n \quad (n > 0)$$

Where;

$D_{upd,n}$ - Updated delay value at n^{th} update

D_{n-1} - Value after $(n-1)^{th}$ update

D_n - Observed value at n^{th} updating cycle

Here, when we observe the first probe vehicle (within a certain time period, for a given demand level) the related value of the initial vector is replaced with value determined with the probe record. And from the 2nd update the weighted average is taken. Note that the basic idea behind this equation is to take the average of all delay values related to a certain event along the route.

Investigation of the System Performances

The network shown in Figure 5 was simulated under different demand levels and different probe sample sizes to investigate system performances under different system and network settings.

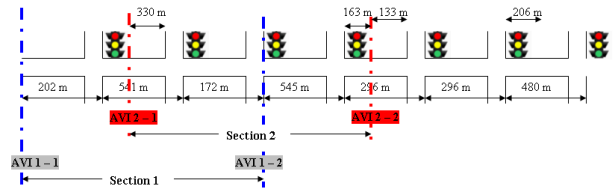


Figure 5 Simulated Network

Here AVI 1-1, AVI 1-2, AVI 2-1 and AVI 2-2 are AVI (Automatic Vehicle Identification) locations. Section 1 and Section 2 are two sections which we selected on the simulated network for examining system performances. Speed limit of the network was set as 55 km/h and numbers of lanes on the route are two for both section 1 and section 2.

First system performances versus different initial settings (discussed under the Section 2) were investigated. Results are shown in Figure 6.

Observing Figure 6, we can determine that the estimated travel times with 'Initially Estimated Vectors' are closer to the reality than other estimates. Regarding 'Initially 50%-50% Vectors' case we can see that the estimate shows a significant deviation from real value, but it gradually gets closer to the real value. Same behavior can be seen in case of 'Initially Empty Vectors'. Therefore it is clear that the system is learning from probe vehicles.

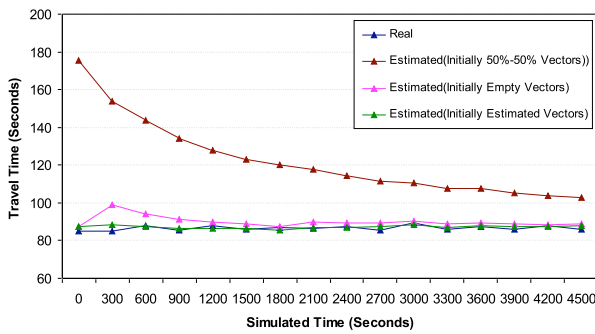


Figure 6 Performance of the System under Different Initial Conditions

Again it is interesting to see the behavior of ‘Initially Empty Vectors’ case. Although initially empty vectors may show a deviation from real travel times, relatively less samples of probe vehicles (within relatively less time period) the system can be trained. Another advantage of ‘Initially Empty Vectors’ is, since those are non-site specific, same system can be used in different sites without re-calibration. Because of convenience of usage and above mentioned advantages and to save the cost for estimating initially estimated vectors it is recommended to use initially empty vectors.

Intuitively we can say that the higher the probe sample size the more accurate the travel time estimation with probe vehicle records. The system performances were investigated under different sample sizes keeping constant demand level (i.e. 1200 vehicles/hour). The system was initialized with initially empty probability and delay vectors. Estimated section travel times were compared as shown in Figure 7.

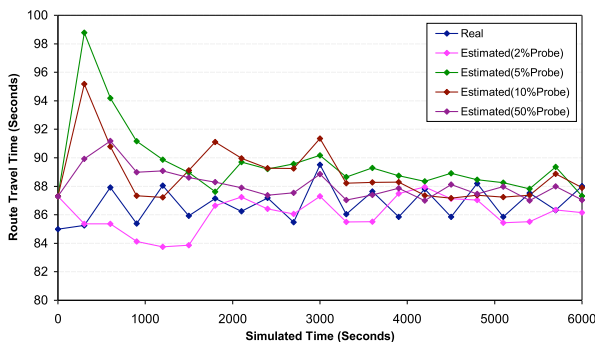


Figure 7 Performances of the System under Different Probe Sample Sizes (Starting Condition: Initially Empty Vectors)

Observing Figure 7 it can be determined that the higher the probe sample sizes the shorter the system convergence time. And further it can be observed that after achieving convergence the system can provide reliable travel time information even with less probe

sample sizes. Figure 8 shows the absolute percentage error along the simulation to give a better insight into the performance of the system.

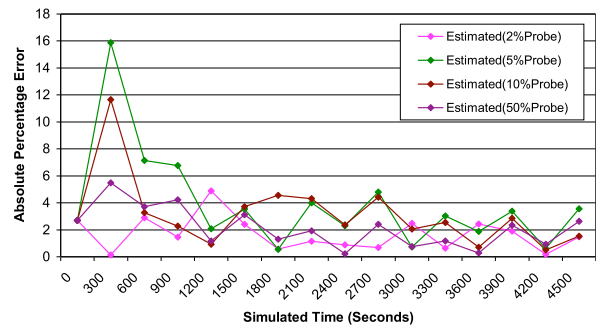


Figure 8 Absolute Percentage Error (Starting Condition: Initially Empty Vectors)

The ability of the system to identify and respond to sudden network changes was investigated. For this analysis at time 0 the signal settings of the first signalized intersection on the Section 2 were changed (Red time=30 Seconds and Effective Green time=30 seconds) and the behavior of the system was analyzed as shown in Figure 9.

Up to time 0, the system has learned from around 6000 probe vehicles (i.e. around 100 hours under constant demand). Therefore at time 0 we have 6000 probe vehicle records.

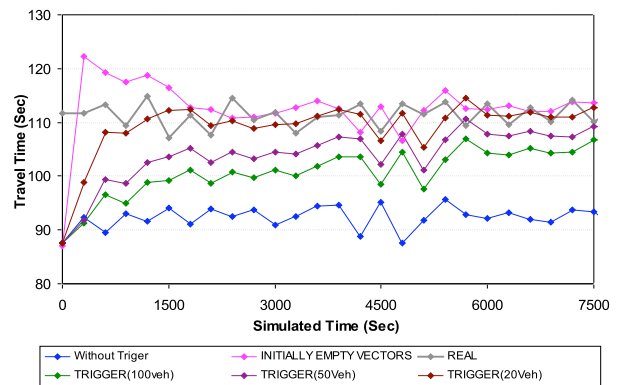


Figure 9 Performance of the System after a Sudden Change in Signal Program

If we update the system without triggering at time 0 as we can see in the Figure 9, it will take a considerable time for the system to adapt to the new signal settings and to achieve the convergence again. But when the system is triggered at time 0, we can see that the system can achieve the convergence within a shorter time period. Here, we are going to trigger the system by changing the database settings. Here “TRIGGER (100Veh)” means at

time 0, hundred (100) hypothetical probe vehicles are created which will provide approximately the same probability and delay vectors at time zero. Then instead of the previous probe records in the database (in this case 6000 probe records) the 100 hypothetical probe records are used to re-initialize the system at time 0.

But the system is triggered (for example every morning) the travel time estimates can be oscillated if there is no any change in the network as shown in Figure 10. On the other hand without triggering it is difficult to understand sudden changes in the network as shown in Figure 9.

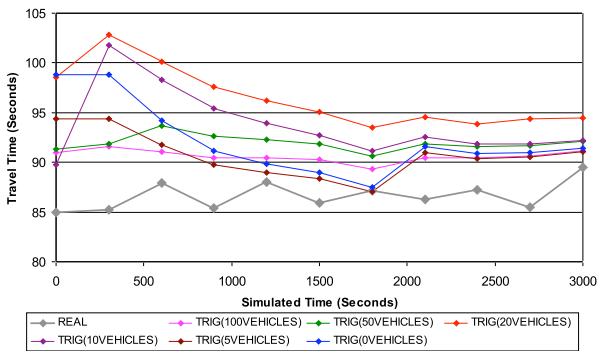


Figure 10 Oscillations of Travel Times after Triggering

Observing Figure 10 we can say that, in case of “TRIG (0VEHICLES)” (i.e. triggering with zero vehicles) although initially travel time estimates show a deviation from real value, it can achieve convergence within a shorter time period. On the other hand it is clear that, when trigger with zero vehicles the system can identify any change to signal settings and can adopt accordingly.

Therefore we propose kind of a *Sub System* in the main system. That is, the main system is continuously learning with probe vehicles and estimates travel times. And we set the sub system in such a way that in every day it starts from initially empty vectors.

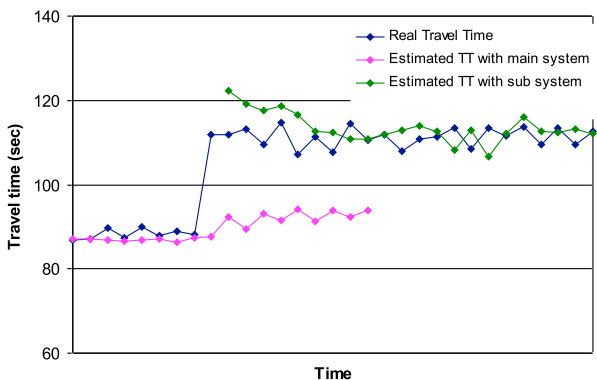


Figure 11 Switching the main system and sub system

The purpose of this *Sub System* is to identify any possible changes in the network. If any change is identified, after pre-defined threshold time period the main system is replaced with sub system.

Figure 11 shows an example for ‘system switching’. Here at the end of certain days (i.e. at the end of the N^{th} day) the signal settings were changed. On the next day (i.e. at the start of the $(N+1)^{\text{th}}$ day) a significant deviation between estimates of the main system and the sub system is observed (see Figure 11) and the system was switched. For this analysis travel times on coordinated section 2 was used under constant demand of 1200 vehicles/hour and 5% of probe.

Observing Figure 11 it can be seen that after changing the signal plan the estimates with the main system and the estimates with the sub system are significantly different. Therefore after 30 minutes the main system is replaced with the sub system. Then the sub system becomes the main system which estimates travel times and learns gradually with probe vehicle records.

4. Conclusions

The *Self Learning Travel Time Estimator* we presented in this paper is a robust system that can be implemented in any kind of urban signalized networks in undersaturated conditions, independent of the geometry and signal settings of the network.

We saw that the self learning ability of the system is independent of initial settings of the system (i.e. initial probability and delay vectors settings) and that the system can achieve convergence and provide reliable travel time estimates. The system was tested under three initial conditions and it was seen that the “Initially Estimated Vectors” have the least convergence time. However, we recommend “Initially Empty Vectors” since the initial cost of calculating those can be saved as well as the system is not needed to be changed when the network settings are changed.

With the introduction of a sub-system apart from the main system in the *Self Learning Travel Time Estimator* we made the system capable of identifying changes in the network such as renewed signal plans etc. It could show to adapt to the new settings.

Although there is a direct relationship between system convergence time and probe vehicle sample sizes, after achieving the convergence there is no significant relationship between the probe sample size and accuracy of travel time estimates. Therefore it can be concluded that the system can provide reliable and accurate travel time estimates independent of probe vehicle sample sizes in the network after achieving the convergence.

With a critical view, next to the achievements above, it has to be in mind that these have just been realized in

undersaturated conditions and with non-adaptive signal controllers. The applicability of the system in over-saturated networks has still to be investigated and new developments to deal with over-saturated conditions have to be added.

Further, in this study, we have ignored the previously stated influence of weather, incidents and other factors. Therefore the system can be extended to incorporate the inter-relationships among events at intersections and possible causes which can affect them such as weather, incidents etc.

It could be argued that under our assumptions a travel time forecast is obsolete, but to understand the behaviour of such an autonomous system, one has to understand the fundamental principal first. In this case, it means that it has to be investigated if such a system would be at all able to recognize patterns in the travel time distribution along signalized urban roads. Now, with a stable and robust base system, we are able to extend the system to decrease the variability in our predictions to provide better and reliable information to the road users. This means, that we understand the system as it is as a base model on top of which we are developing new tools and functionalities.

Further, we have to run validations with real world data to back up our simulation based tests. So far we had too few data to investigate the convergence of the system, but are convinced that future datasets will enable us to prove the convergence in different places and situations.

5. References

Fernandes, C.M., An Algorithm to Handle Structural Uncertainties in Learning Bayesian Network, Brazil, 2004.

Ruiz, C., Illustration of the K2 Algorithm for Learning Bayes Net Structures, *Lecture Notes on Machine Learning*, Department of Computer Science, Worcester Polytechnic Institute, Massachusetts, 2005 Spring.