Margdarshak: A Mobile Data Analytics based Commute Time Estimator cum Route Recommender

Rohit Verma, Aviral Shrivastava, Sandip Chakraborty, Bivas Mitra

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, INDIA 721302
{rohitverma.kgp, aviralshrivastava93, sandipchkraborty}@gmail.com, bivas@cse.iitkgp.ernet.in

ABSTRACT

Waiting at traffic signals and getting stuck in traffic congestion eats a lot of time for a commuter in most of the metro cities of the world. Although there exists a large pool of navigation applications, but all of them turn out to be ineffective for dynamically finding out the best route under uncertainty. In this work, we present Margdarshak, a navigation system which utilizes the impact of congestion and wait time at traffic signals for estimating the travel time over a route. We collected a month-long traffic data from different routes at five various cities in India for analyzing the problem in detail. The evaluations performed over the system show that Margdarshak gives a mean estimation error of ± 1.5 minutes, and performs significantly better under uncertainty, compared to other state of the art navigation systems like Google Maps, Here Maps and Waze.

Keywords

Navigation; Traffic Signals; Congestion

1. INTRODUCTION

The large population and heavy traffic in big cities makes the travel time estimation to reach destination for the city commuters a troublesome job [8]. In spite of all estimations and precautions taken to be on-time, more often people meets failure because of an unforeseen and unpredicted road traffic condition. An eye to this problem, which assails a large pool of commuters during their regular travel on city roads, should therefore undoubtedly go in the planning of a smart city. The road traffic in big cities faces two problems - 1) unpredictable road congestions during the busy hours, and 2) waiting time at the traffic signals, which may vary based on the traffic condition. Moreover, sudden congestion in one road segment may trigger cascading effect to the other road segments, and traffic signals [13]. Although the previous studies [10,12–14] develop various mathematical models for pre-estimating average delay at traffic signals and the impact of traffic congestions, however, real time monitoring and estimation of delay under such uncertainty is still a challenge. Moreover, dynamic and real-time route recommendation, considering such uncertainty, can be of immense help to the daily commuters of a big city. The growing usage of smart-phone navigation apps opens up new direction of solutions to this problem, which we explore in this paper.

WPA'16, June 25-30, 2016, Singapore, Singapore
© 2016 ACM. ISBN 978-1-4503-4328-2/16/06...$15.00
DOI: http://dx.doi.org/10.1145/2935651.2935655

State of the art: An ideal navigation system would be the one which adapts the user's current route and accordingly makes dynamic adjustment of the travel time estimation. It is affected by several factors, major being the traffic. On an average, a person in Washington DC annually spends 82 hours in traffic [4]. There are several challenges in estimating travel time while considering the traffic. The effect of traffic congestion, being the major challenge for travel time estimation, has been addressed by many state of the art approaches that utilizes smart-phone or sensor based technologies. CrowdITS [9] uses crowd-sourced global positioning system (GPS) information from different smart-phones along with annotations done by users to report events like traffic congestion, accidents etc. However, it requires user involvement which is a major limitation of their system. Similarly, [7] is a community based navigation application which utilizes manual responses for its navigation process. On the other hand, SMaRTCaR [11] uses off-the-shelf sensors and dedicated modules to get traffic information in real time. VTrack [16] uses different sensory information, like Wi-Fi signals, and performs a map matching procedure using hidden Markov model to estimate the travel time while considering traffic congestion. The work by Skabardonis *et al.* [15] uses several detectors on roads, GPS data from cars and the traffic signal information to estimate the travel time. However, all these approaches and even the most commonly used Google Maps lack the support of taking the effect of dynamic wait time at traffic signals into account, which is quite uncertain during busy hours as pointed out in [13]. Several news articles, such as [5], often talk about the long waiting on traffic signals which leads to huge delays in reaching the destination. Although, [17] utilizes multi-agent reinforced learning algorithms to learn the functionality of traffic light controllers and hence estimate the expected waiting time at the signals, it does not consider the uncertainty during busy hours, or the back-pressure cascading due to traffic congestion.

Limitations: In summary, we identify several limitations of the state of the art, which are enumerated below.

- Most of the systems do not consider the dynamics of wait time at traffic signals. Either they consider past history, or rely on some mathematical estimation. Nevertheless, real time monitoring of current information is required for accurate estimation of travel time.

- Existing real time monitoring systems depends on manual annotation for accurate navigation [7,9].
- They use of off-the-self sensors [11] which required to be pre-deployed. This increases infrastructural cost.
- The impact of congestion on travel time is considered, only when the vehicle itself gets into one. However, they do not consider the cascading effect on waiting time.

Contribution: This paper develops *Margdarshak*, an unobtrusive travel time estimation system, which senses the real time traffic condition from the crowd-sourced GPS trails of the users. Moreover, the system recommends the best route to take from source to destination location, based on the current traffic condition; this recommendation is real time in nature. The core of the system is driven by two key components (a) the accurate estimation of wait time at the traffic signals (b) delay in the road segments due to traffic congestion and its cascading effects. It is important to note that the travel time estimation is a dynamic event; the estimation varies with the change in the traffic condition. Moreover, we conduct two months long study in five metro cities in India - Kolkata, Bengaluru, Hyderabad, Chennai and New Delhi, to evaluate how the commuter navigation is affected by the congestion at traffic signals. Based on this study, we develop a database of 2 routes of each of the cities – Kolkata, Bengaluru and Hyderabad, and 1 route each at New Delhi and Chennai, for evaluating *Margdarshak* system. The system gives a mean error of $\pm 1.5 minutes$ over a trail.

Organization: The rest of the paper is organized as follows. Section 2 describes the motivation behind the work followed by the overall system overview as discussed in Section 3. A detailed description of the system is given in Section 4. We then evaluate the system and analyze the performance results in Section 5. Finally Section 6 concludes the paper.

2. MOTIVATION

In this section, we conduct motivational experiments to highlight the limitation of the commercial travel time estimation tools. In that line, we uncover the different factors which results in the erroneous travel time estimation. We conduct experiments in the cities of Kolkata and Bangalore for one month, with three types of devices namely Moto G2, Nexus 4 and Asus Zenfone5. This experiment covers in four routes in these two cities, covering total of 124km.

How accurate are the existing tools? We perform the experiments with two commercial and popular navigation tools - (a) Google map [2] and (b) Waze [7] for travel time estimation. First we evaluate the performance of the two aforementioned systems against actual travel time in four different routes. Figure 1 demonstrates the result. We observe a difference of 8-9 minutes with actual travel time in different routes, sometimes (say R-4) the error shoots up to 30 minutes. This error may jeopardize one's travel plan considerably. The error gets more prevalent with Waze. Close inspection reveals that, most of the vehicles get stuck in congestion or at traffic signals, which results error in travel time

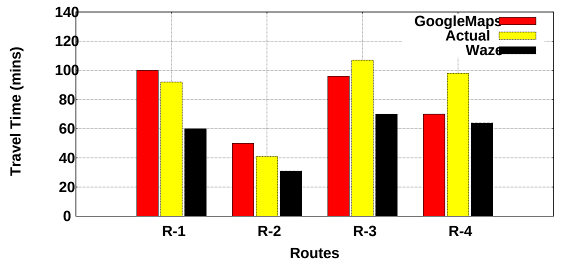


Figure 1: Travel Time Estimation

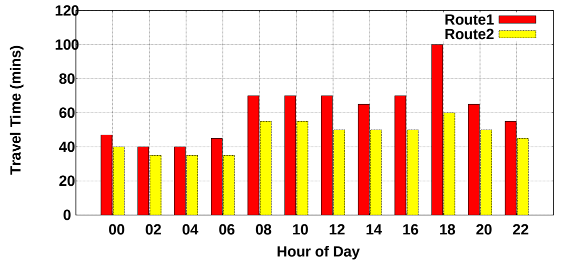


Figure 2: Travel Time Trend over a Day

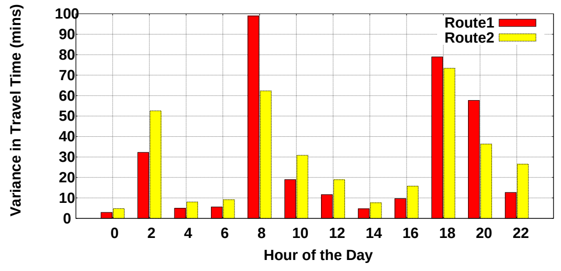


Figure 3: Trend Over Multiple Days

estimation. This points to the fact that, accurate waiting time estimation at different traffic signals may substantially improve the estimation of commute time.

What is the impact of time of a day? We explore the accuracy of travel time estimation at different time of the day. Figure 2 illustrates the trend of travel time over a day for a two hour interval for two routes. This is intuitive to observe that in early morning, the traffic signal and congestion do not incur much error due to low traffic. However, travel time is high in office hours, between 8 am to 6 pm, with 6 pm showing the peak. This time window would also show the high impact in congestion and delay at traffic signals, resulting incorrect estimation of commute time.

Does past data help? Several existing strategies [10,12, 14,17] make use of past data to predict travel time. However, Figure 3 reveals that, past data fails to predict travel time in many cases. The plot shows the travel time variance for two different routes, over a full day from Mon-Thu in a week. We observe that the travel time variance is as high as 100 minutes in the same route.

3. SYSTEM OVERVIEW

Figure 4 shows different components of Margdarshak, which follows a client-server architecture. Database is generated with the help of war-driving data with annotated traffic signals. The server-side component implements this processed database and the navigation modules, whereas the client-side implements the online module running on the smart-phone, which is responsible for initialization of a trip and crowd-sourced data collection.

3.1 War-driving

The proposed system estimates the travel time and determines the best travel route from the (a) past data and (b) the current state of the car movement (inferred from the smart-phone sensors, refer Section 3.3). In order to get the past data of the route information, we relied on the war-driving approach. We develop a smart-phone app to collect the route data, where we log the time-stamped GPS traces for different city routes. The volunteers covered 8 different routes, at the city of Kolkata, Bangalore, Hyderabad, New Delhi and Chennai. During the data collection, they also manually annotate the locations of the traffic signals on the route. The collected data is processed to develop the repository of the routes.

Database Generation: The core of the database is a table called *Traffic Signal Details Table* which stores the complete route details. Each row stores the next traffic signal ID in the current route, the previously encountered signal ID the distance between these two signals, and the wait time associated with the next traffic signal. However, one traffic signal may be part of multiple routes, and we would have to store duplicate signal specific information in this table. Moreover, there can be multiple routes, hence we need to store the route details multiple times; this increases redundancy in the table. Hence we take help of the following two auxiliary tables to normalize the database, (a) *Traffic Signal Table* and (b) *Route Table*. The *Traffic Signal Table* stores all the signal locations and their IDs. A particular signal can be a part of multiple routes, and therefore we also need to separate the routes. The *Route Table* stores the information of a specific route, i.e, the *route ID* and *route name*.

3.2 Server Side

Server side implements the navigation infrastructure comprising of database processing and navigation. The details follow.

3.2.1 *Processing of Database*

The most crucial and expensive operation of the system is the continuous polling to the database since parallel queries from different clients will slow down the system considerably. In order to ensure faster access during navigation, we store the database in a directed weighted graph, called a *Route-Signal (RS) Graph*. Each node N_i in the RS graph represents a traffic signal i. The edge $E_{i,j}$ between the two signals i and j contains the weight as the distance between those signals. Hence, the database is queried only at the start of the trip and once the RS graph gets generated, all the operations are performed on this graph. From the database, the RS graph is generated by (i) joining the *Traffic Signal Table* and *Route Table* to get route specific signals then (ii) further joining with *Traffic Signal Details Table* to get the right order of signals along with details. It is important to note that, the RS graph can also be automatically extracted from the city map (using [1] and the assumption that there

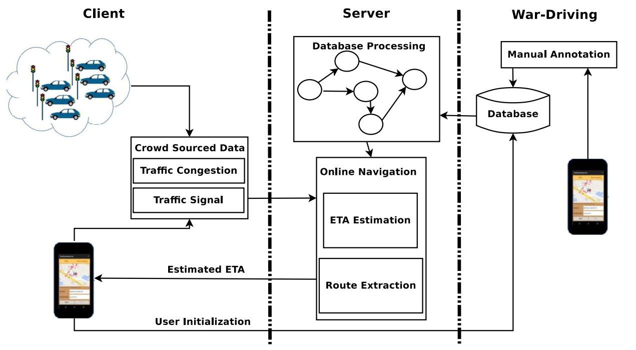


Figure 4: System Architecture of Margdarshak

would be a traffic signal at every road crossing), when we do not have war-driving data with manual annotation of signals; this may mitigate the cold start problem.

3.2.2 *Online Navigation*

The navigation module discovers the best route between a source-destination pair, that provides minimum estimated time of arrival (ETA). The ETA denotes the time to reach the destination following a feasible route, and needs to be dynamically updated based on the current road traffic. However, during the run-time, the recommended travel path may get changed based on the real time ETA information. ETA has two major components:

- A fixed time component based on the speed of the vehicle and distance. This takes into account the time taken from source to the first signal, time taken to reach the last signal from the first signal and then from the last signal to the destination. Note that most of the wait time variability is in between the first signal and the last signal, which is captured in the next component.

- A dynamic time component which is calculated on the basis of the wait time at signals and any delay because of congestion. A traffic congestion may result in cascading delay in wait time at the signals.

3.3 Client Side

The client side of the system, running on the smart-phone, implements the two modules: (a) crowd-sourced data collection, and (b) user interface. Data collection module collects sensor data from the smart-phone and gets the annotation input from the user (in the form of app). The output of this module from several such app users is crowd-sourced by the server to construct the database and the *RS graph*. User interface (mostly relies on the Google map) extends the facility such that a user can interact with the system by providing her source and destination information. It also provides the display such that she can get the the recommended route and its ETA.

4. METHODOLOGY FOR MARGDARSHAK

In this section, we propose the methodology for accurate ETA estimation and dynamic route recommendation. It has three components – (a) user initialization – where commuter initiates the process, (b) online navigation – where system estimates the ETA and recommends the best route, and (c) the user interface to interact with the commuters.

4.1 User Initialization

The user, once starting the system, provides the source-destination information to the client module. This is the only time that the database is polled; immediately the *Route-Signal (RS) Graph* gets generated from the table. Now every successive processing is done on the *RS graph*.

As mentioned earlier, the dynamic time component is computed based on the wait time prediction at all the traffic signals in the route. Therefore the system needs to locate the traffic signals that one may encounter on the probable routes. Next, the system extracts the probable routes from the database and annotates the signal information for those routes. Therefore in the initialization process, one major challenge is to locate the nearest traffic signal succeeding the source location and the nearest preceding signal of the destination location. Finding the closest signal may not solve the problem because the preceding signal (instead of succeeding) can also be the closest one for the source location; same might occur with the destination location. This issue is resolved using the following methodology:

1. Convert the signal positions to *Universal Transverse Mercator* (UTM) [6] plane following the standard procedure. Let the sequence of signals on a route be $\{S_i, S_j, S_k, ...\}$

2. Start with the first two traffic signal pairs S_i and S_j on the route (first signal and the succeeding one) and obtain the equation of the line.

3. Project the source location co-ordinate T on this line. If this location lies on the line segment, in between the two above points, then the traffic signal S_j is the required succeeding signal. Otherwise try with the next pair of signals, say S_j and S_k.

4. Try this for a set of pairs within a given distance from the source. Otherwise, take the first signal S_i as the succeeding signal.

5. Follow the same process for destination; In this case the preceding signal is the required one.

Illustration: Consider an example scenario in Figure 5. The user selects the source location T at a point between signals S2 and S3. The system has the sequence of signals {S1, S2, S3, S4, S5, ...} stored in the database for a probable route, and the server needs to identify the nearest succeeding signal for the starting position T. If the closest signal gets selected, then S2 will be chosen as the succeeding signal, which is erroneous. Hence, following the aforesaid algorithm, we draw the line L1 passing through signals S1 and S2. Then we project the point T on the line. It is easily observable that the point T wouldn't lie on the line segment between S1 and S2. Hence, we move on to the line L2 between S2 and S3. We then project the point T on the line segment created by L2. This time T lies on the segment line L2 and L3, hence, signal S3 gets selected as the succeeding signal of source T.

4.2 Online Navigation

Once the user provides the source-destination location, the next task is to discover the best route dynamically.

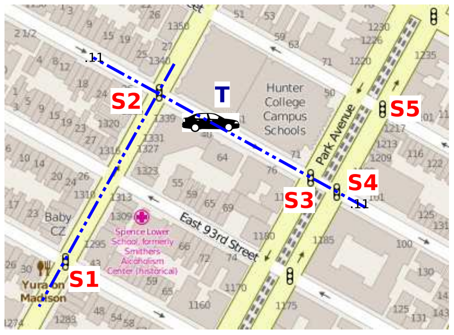


Figure 5: Deciding which signal to select. The system has the sequence of signals {S1, S2, S3, S4, S5} stored in the database. Draw the line L1 passing through signals S1 and S2. Projected point of T doesn't lie on line segment between S1 & S2 but lies on the line segment between S2 and S3 of line L2. Hence, signal S3 gets selected as the succeeding signal of source T

Whenever the user arrives at a signal, the system recalculates the route based on the ETA. This is basically done by estimating the ETA across all the possible routes (from the *RS graph*) and selecting the best one. As mentioned earlier, the ETA estimation takes into account two factors into consideration: (a) the fixed component of wait time at traffic signals, and (b) the dynamic wait time component due to traffic congestion. Next we discuss how these two aspects are handled.

4.2.1 Fixed Wait Time Component at Traffic Signals

The fixed time component is computed from the past history by finding out what is the average wait time at every signals on a route. However, considering outdated information may result error in estimation, as the wait time has correlation with the day as well as the time of the day. Considering these facts, we split the 24 hours of a day into a number of fixed time-zones, and the first vehicle in a time zone, that provides crowd-source information about the wait time, is called the bootstrapping vehicle. The bootstrapping vehicle helps in the initial calibration of wait-time at the signals, and the information collected from subsequent vehicles contribute to improve the accuracy. Accordingly, we consider the following points during estimation of the fixed wait time component at different traffic signals.

- By default, a signal has an associated wait time Γ stored in the database. The bootstrapping vehicle helps to calculate the extra τ wait time that the signal has due to congestion. Also, τ can be negative in cases when signal is clear.

- Each signal thus has a time $(\Gamma + \tau)$ linked to it, which is further refined (updated in the database) by crowd-sourcing through several vehicles passing by. This updated time is used to calculate the new dynamic time of a trip.

Illustration: Assume that in a trip, a vehicle encounters a series of signals $\{S_1, S_2, ..., S_i, ..., S_n\}$ between the source and the destination. When a bootstrapping vehicle starts, say it takes a time T to travel from the source to S_1. At S_1, say it waits for an additional τ_1 time apart from the normal wait time Γ. Then we can calibrate this signal's wait time as $\Gamma + \tau_1$. This would continue for the other signals too.

So, eventually the wait time at each signal i can be given as $\Gamma + \tau_i$, where $\tau_i \geq 0$. It should be kept in mind that if there are n signals on a route its not necessary that all these signals are bootstrapped using a single vehicle. Multiple vehicles can bootstrap parts of the route. Now, whenever a new vehicle arrives, it has the information of τ_i at each signal (which can now be used for better estimation) and might further get updated in some scenarios.

4.2.2 *Dynamic Wait Time Due to Congestion*

Congestion is detected by observing the speed and location change of a vehicle in a small area. The system checks if the vehicle is having very low speed (computed from accelerometer readings), and also if the location is not changing much, and also this is the same for other vehicles in that region. Congestion can be observed at a traffic signal or at any other location. The one at traffic signal will be taken care by the same module as discussed earlier. The latter case would decrease the speed of vehicles by a factor, which would reflect on the travel time. This dynamic factor of reduction of speed is taken into account for updating the dynamic time due to congestion in areas other than traffic signals. This factor is calculated as the mean over all the vehicles in that area. Many of the current applications do take into account the impact of congestion on the travel time but this impact is only effective if the vehicle itself gets into congestion. Our system continuously runs a congestion detection thread and finds the impact on the travel time of the vehicle by modifying the speed for the congested region while calculating the travel time.

Once the fastest route is selected, navigation starts and the ETA gets updated every second. The system is recalibrated at each signal on the route. Further, any occurrence of congestion is accounted for, as and when it happens. A background thread always keeps on checking the possibility of congestion on the route.

4.3 User Interface

On launching the application, the user gets a map where she can select the source and the destination. Once the system gets this input, it estimates the time taken on all the possible routes and selects the fastest one by default, which the user can change. Once the navigation starts, the travel time is decreased with respect to the vehicle speed and is recalculated at each signal.

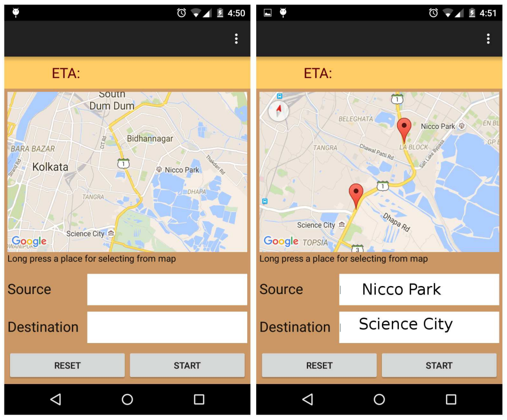
5. PERFORMANCE EVALUATION

We conduct experiments to evaluate the performance of Margdarshak for eight different routes across five metro cities in India – two routes each in Bangalore, Kolkata and Hyderabad and one each in Chennai and New Delhi. In our experiments, the volunteers traveled on these routes, and used the app for a week at different time of the day. The collected data is used for analyzing the performance of the system. We compare our system with Google Maps [2], Waze [7] and Here Maps [3]. In this section, we explain in detail the experimental set up and performance of Margdarshak.

5.1 Experimental Results

5.1.1 *Competing Heuristics*

The key feature of Margdarshak's efficiency is its dynamic updates on travel time estimation on the basis of traffic congestion and wait time at traffic signals. Figure 7 represents the travel time estimation for different routes by different state of the art platforms mentioned above. Clearly, Margdarshak outperforms other platforms with a significant difference in travel time, as Google Maps and Here Maps highly rely on past data which leads to error prone travel time estimation. In addition to that, Waze also involves human annotated events, which may or may not be accurate at any instance of time.



Source	
Destination	

Source	Nicco Park
Destination	Science City

Figure 6: Application Screen-shot

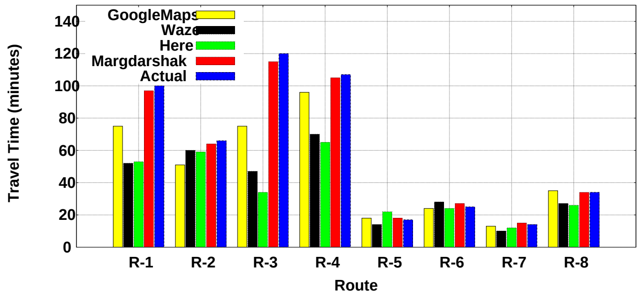


Figure 7: ETA comparison across different routes; Kolkata (R-1, R-2), Bangalore (R-3, R-4), New Delhi (R-5), Chennai (R-6), Hyderabad (R-7, R-8)

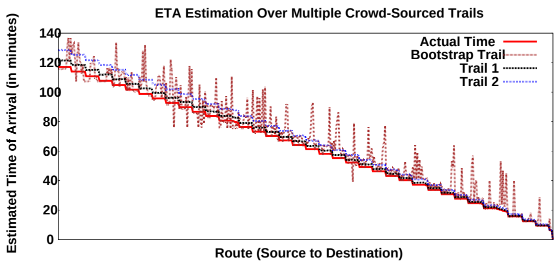


Figure 8: ETA on Crowd-sourcing in Bangalore

5.1.2 *System Evaluation*

We evaluate our system on the basis of two aspects as follows.

Improvement in accuracy through Crowd-sourcing: Margdarshak takes full advantage of crowd-sourced GPS data to improve its accuracy in a timely manner. Figure 8 shows travel time estimation in different scenarios on a specific route in Bangalore. The horizontal axis depicts the travel from source to destination on that route. Along with the actual travel time, we show the estimated travel time in the cold start scenario (the bootstrap trail, when there is no prior information), trail 1 (sometime after the bootstrap trail, when few vehicles have passed on that route and provided information via crowd-sourcing) and trail 2 (sometime after trail 1, we have information from more vehicles). As we can see from the figures, although the error rate is higher during the cold start, but it reduces rapidly as soon as more number of vehicles travel through the same route. The estimated value for trail 2 is significantly closer to the actual travel time.

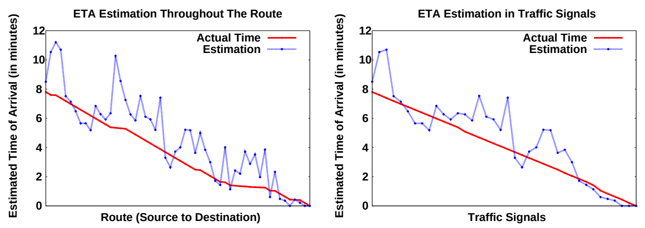


Figure 9: Estimated Vs Actual travel time comparison (a) throughout the route, (b) at traffic signals

Travel time estimation accuracy: Finally, Figure 9 further concretes the claim of high accuracy in travel time estimation. We show the actual travel time and the estimated travel time for two cases – (a) the travel time is updated at every second throughout the route, and (b) the time is calibrated at every signal. The figure shows that the mean error of travel time estimation is ±1 min at the traffic signals and ±1.5 min throughout the route, whereas the maximum error is 3.5 min at signals and 4.9 min throughout the route. This indicates that dynamic update of routes based on the crowd-sourced information of waiting time at the signals can make actual travel time closer to the estimated travel time. It should be noted that the mean travel time error is low at the traffic signals because of recalibration. Every time, the error starts accumulating, it is reduced at the signal and so the low error.

6. CONCLUSION

The problem of estimating travel time has been targeted several times and versatile solutions have been made available for the same. However, they have their own limitations as most of them do not consider dynamic re-routing under uncertainty. Some rely on historical data, whereas some require user annotations or external hardware to tackle the problem. In this paper, we presented Margdarshak, a dynamic, crowd-sourcing based travel time estimation system for private vehicles, which estimates traveling duration on the basis of real time traffic congestion and wait time at traffic signals. Several experiments conducted over different cities, while comparing with existing systems like Google Maps, Here Maps and Waze, show that Margdarshak stands ahead of all with promising results. The system delivers a mean error of $\pm 1 minute$ on traffic signals while that of $\pm 1.5 minutes$ throughout the route.

Acknowledgement

The authors would like to thank Information Technology Research Academy (ITRA), Department of Electronics and Information Technology (DeitY), Government of India, for supporting this work under the research project "Post disaster situation analysis and resource management using delay tolerant peer to peer wireless networks (DISARM)", sanction letter number and date ITRA/15(58)/MOBILE/DISRAM/01, Dt. 19-09-2013.

7. REFERENCES

[1] CGTK GPX Generator. http://gpx.cgtk.co.uk/Oqdxr.
[2] Google Maps. https://maps.google.com.
[3] Here Maps. https://maps.here.com/?map=22.3333,87.3333,10,normal.
[4] Reuters. http://www.reuters.com/article/us-usa-traffic-study-idUSKCN0QV0A820150826.
[5] Urban Mobility Scorecard Annual Report. http://inrix.com/scorecard/.
[6] UTM. https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system.
[7] WAZE. https://www.waze.com/.
[8] Wikipedia, Traffic Congestion. https://en.wikipedia.org/wiki/Traffic_congestion.
[9] Ali, K., Al-Yaseen, D., Ejaz, A., Javed, T., and Hassanein, H. S. Crowdits: Crowdsourcing in intelligent transportation systems. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE* (2012), IEEE, pp. 3307–3311.
[10] Arnott, R. A bathtub model of downtown traffic congestion. *Journal of Urban Economics 76* (2013), 110–121.
[11] Campolo, C., Iera, A., Molinaro, A., Paratore, S. Y., and Ruggeri, G. Smartcar: An integrated smartphone-based platform to support traffic management applications. In *Vehicular Traffic Management for Smart Cities (VTM), 2012 First International Workshop on* (2012), IEEE, pp. 1–6.
[12] Comert, G. Simple analytical models for estimating the queue lengths from probe vehicles at traffic signals. *Transportation Research Part B: Methodological 55* (2013), 59–74.
[13] Heidemann, D. Queue length and delay distributions at traffic signals. *Transportation Research Part B: Methodological 28*, 5 (1994), 377–389.
[14] Lindsey, C. R., van den Berg, V. A., and Verhoef, E. T. Step tolling with bottleneck queuing congestion. *Journal of Urban Economics 72*, 1 (2012), 46–59.
[15] Skabardonis, A., and Geroliminis, N. Real-time estimation of travel times on signalized arterials. Tech. rep., 2005.
[16] Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., and Eriksson, J. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems* (2009), ACM, pp. 85–98.
[17] Wiering, M., et al. Multi-agent reinforcement learning for traffic light control. In *ICML* (2000), pp. 1151–1158.