Physical and Virtual Cell Phone Sensors for Traffic Control: Algorithms and Deployment Impact

Shumeet Baluja
Google Research
shumeet@google.com

Michele Covell
Google Research
covell@google.com

Rahul Sukthankar
Google Research
sukthankar@google.com

***Abstract*—Decades of research have been directed towards improving the timing of traffic lights. The ubiquity of cell phones among drivers has created the opportunity to design new sensors for traffic light controllers. These new sensors, which search for radio signals that are constantly emanating from cell phones, hold the hope of replacing the typical induction-loop sensors that are installed within road pavements. A replacement to induction sensors is desired as they require significant roadwork to install, frequent maintenance and checkups, are sensitive to proper repairs and installation work, and the construction techniques, materials, and even surrounding unrelated ground work can be sources of failure. However, before cell phone sensors can be widely deployed, users must become comfortable with the passive use of their cell phones by municipalities for this purpose. Despite complete anonymization, public privacy concerns may remain. This presents a chicken-and-egg problem: without showing the benefits of using cell phones for traffic monitoring, users may not be willing to allow this use. In this paper, we show that by carefully training the traffic light controllers, we can unlock the benefits of these sensors when only a small fraction of users allow their cell phones to be used. Surprisingly, even when there is only small percentage of opted-in users, the new traffic controllers provide large benefits to all drivers.**

I. Introduction

One of the largest complaints of commuters in the Mountain View, California area is the amount of traffic they face during the morning and evening rush hours. One of the problem areas, controlled by seven main lights, is shown in Figure 1. The goal of our project is to evaluate the timing of the traffic lights on these intersections and also improve them through better control algorithms and improved sensors. In [6], we first addressed the problem of creating improved control algorithms for the traffic light logic. In that study, we assumed that the standard induction-loop sensors would be used. In this paper, we examine the possibility of replacing induction-loop sensors with sensors that monitor cell phone signals as a proxy measure for local traffic density.

Induction loop sensors are currently widely deployed, and have been in use since the 1960s. Induction loop sensors are placed inside the roadway's pavement and, at a high-level, work by creating an electromagnetic field around the loop area. As vehicles enter and exit the field, fluctuations in the field are recorded as an indication that a car has passed over [5].

Despite being the most common type of sensor, other road-mounted detectors such as cameras and microwave sensors are also employed [20] [23]. Each provides information about the number and type of vehicles on the road, their speed,

Fig. 1. Area to be optimized around Mountain View, California. A rush hour traffic flow shown.

and travel times, etc. They, like induction loops, can require significant roadwork for deployment, frequent maintenance and monitoring, and are sensitive to proper repairs and installation work. Further, because of the variability in climate and traffic conditions, construction techniques, materials, and even surrounding unrelated ground work are often cited sources of failure. Good overviews of failure modes and rates (reaching above 25% in some geographies) can be found in [5] and the Department of Transportation studies [8].

Instead of induction-loop sensors, directional sensors which scan for the presence of a car by a passenger's cell phone have the potential to eliminate the need for placing sensors in-ground and for alleviating many of the associated monitoring, maintenance, and failure issues. To detect whether cars are passing by, these sensors can use signals beyond the typical radio-signals emanating from older cell phones, and use radio frequencies such as wi-fi and bluetooth commonly found in

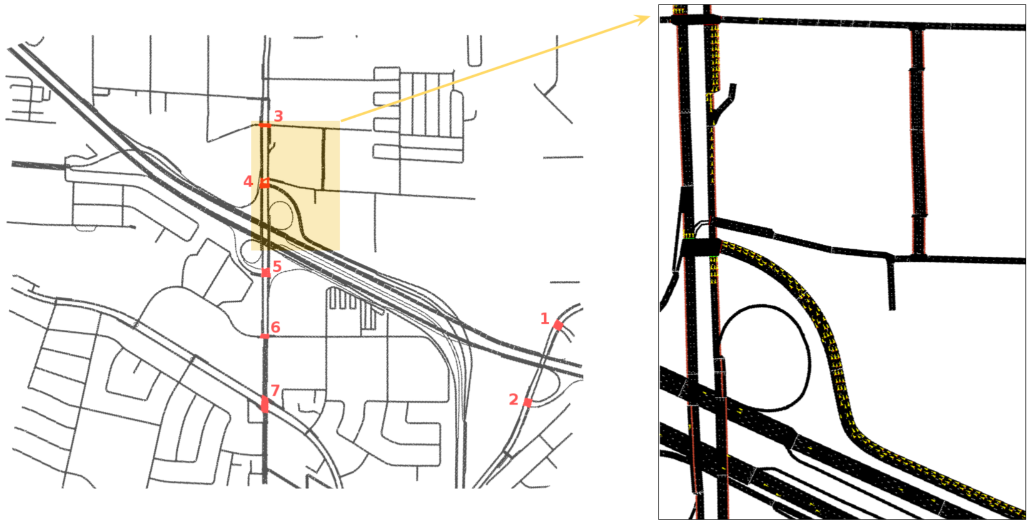

Fig. 2. Roadway data imported into the SUMO simulator. Left: The 7 lights to be optimized. Right: expanded region of traffic; yellow triangles shows a representative traffic scenario.

smart phones today. See [19] for a good overview of cellular phone detection techniques (a sample commercial cell phone detector for use in vehicle and pedestrian applications can be found in [14]). An alternative to deploying physical sensors is to use *virtual sensors* that effectively work in the same manner as physical ones. Through applications that reside on cell phones, such as navigation/map applications that use GPS as well as tower triangulation, drivers' geo positions and velocity can be determined and fed into a centralized server that controls light controllers. Though not currently typically done, this becomes possible if light controllers are given access to external real-time data sources (a central server) that aggregates anonymized driver positions.

Physical and virtual sensors each have their benefits. The largest benefit for physical sensors is that traffic lights do not have to be constantly connected to a central server. Rather, the physical cell phone sensor can be deployed in a manner similar to the already well understood locally connected induction loop sensor. On the other hand, virtual sensors have the benefit of not deploying extra hardware once the traffic lights are connected to a centralized controller. Additionally, a variety of supplementary information gathered through a cell phone's multitude of sensors can also augment the GPS tracks. Further, virtual sensors have the added benefit of easily being able to use space-time correlations in travel tracks to eliminate potential over-counting of traffic in scenarios in which multiple people in the same vehicle are each using their own cell phones. Additionally, more sophisticated intelligence algorithms, such as machine learning classifiers that differentiate pedestrian and bike movement patterns from automobile traffic can be deployed and updated easily. Though still possible in the case with physical sensors, eliminating the unwanted signals will either require increased sensor sensitivity or longer term information across multiple sensors.

Whichever type of sensor is used — physical ones that interact directly with the radio signals or virtual ones that require access to user transmitted data through cell phone-based applications, there must be social acceptance of the use of this data for traffic lights. Though absolutely none of the signals require the identification of a user, or strictly necessitate knowing that the same (anonymous) user was at two intersections, users need to become comfortable with municipalities having access to even this level of anonymized data. Privacy concerns abound; see for example [1] [12] [15] [18]. Additional worries, some more real than others, over battery usage, compatibility, fairness, and other policy decisions must all be proactively addressed. Many issues may lead to slow deployment of these sensors.

This leads to a chicken-and-egg problem. Without being able to demonstrate the benefits of using cell phone sensors (virtual or physical), wide acceptance of the idea may be slow. What can we do in the meantime? In this paper, we show that by carefully training traffic light controllers, we can unlock the benefits of these sensors when only a small fraction of users participate. Even when the adoption percentages are small, properly trained traffic light controllers that employ cell phone based sensors can reveal large benefits *to all drivers* — even those that choose not to opt-in.

II. Real-World Data

This section describes the data we collected for the experiments presented in this paper. For real-world traffic experimentation, two sets of data are needed: roadway information (layout, speed limits, etc.), and travel track information: time-stamped, anonymous location of users in the region of interest. Note that the travel tracks are just used for training and testing the controllers; they are not required for deployment.

To gather the road information, we combined the data available from Google maps and OpenStreetMap [10]. The results provided reasonable estimates of roadway positions as well as traffic light locations, as shown in Figure 2. The figure shows the maps as rendered by the traffic simulator, SUMO [13] (SUMO is open-source and can be freely downloaded [7]).

In addition to accurate road information, demand for each road section must be modeled. We created a demand profile through anonymized location data collected from opted-in Android cell phone users [3]. The data was collected over several months. The raw data, which itself does not include personally identifiable information (PII), was also scrubbed to further reduce identifiability risks.

From this data, we select data with travel-tracks that intersect with the map area shown. We also filter by time, limiting to looking at a given start and end date — in particular around rush hour periods. We then filter all the given times down to the weekday and time of day (e.g., Tuesday 7am local time). This provides a close-to-realistic profile of the road demand for Android users. By *overlapping* the gathered data over weeks, we compensated for the fact that all travelers are not Android users who provide their data.

III. TRAINING TRAFFIC LIGHT CONTROLLERS TO USE CELL PHONE-BASED SENSORS

Many different types of light controllers are currently deployed. The simplest is the static light controller, in which there is no communication with other lights or sensors. The light cycles between phases at predetermined intervals. On the other end of the spectrum, more sophisticated experimental controllers incorporate communication with other lights and/or cars, and employ planning-based approaches [22]. Between these two extremes are reactive controllers that use induction loops to sense when a car is waiting and adjust their phase switching timings based on waiting traffic. How and to what extent the sensors influence the controller's decision varies and continues to be an active area of research. In a recent study, [6] cast the controller design problem into a micro-auction based framework in which each phase *bids* for a turn to become activated (*i.e.,* if a sensor indicates that the queue to turn left is suddenly non-empty, the phase with a left turn arrow will effectively increases its bid to attempt to get instantiated.) [1] In that work, the controller relied on inputs from typical induction-loop sensors and achieved promising results in two cities under varying heavy traffic loads. In this paper, we will use exactly the same controllers with cell phone-based sensors to test the effectiveness of these new sensors, while keeping the underlying scheduling algorithms/controllers the same as those studied previously.

Because cell phone sensors can be wirelessly deployed without significant road work, it is possible to use multiple sensors instead of the usual single induction loop that is typically placed in the pavement next to a traffic light. In our simulations, we replace each induction loop sensor with three cell phone sensors — placed 30, 60 and 90 meters away from the light along all the (typically 4) directions of the intersection. For our simulations in Mountain View, California, this was done for seven intersections. Unlike induction loop sensors, however, these sensors can only detect a fraction, F, of the cars. That fraction is a controllable parameter and is set to the percentage of cell phone users that we expect the sensor to detect (either are able to detect or that users give permission to detect). As will be described in the next section, we vary F from 0% to 100% in testing to model different uptake rates. We also explore the possibility of varying F during training; the hope is that the controllers will learn to use the sensors when appropriate and ignore them otherwise.

No matter which algorithm/controller-type is used for traffic light control, each approach has numerous parameters that must be specified to complete the program. Even in the simplest controller with fixed schedules, for each light, the length of the phases and their offsets have a large impact on the performance of the system. Numerous machine learning approaches have been used in setting the parameters and algorithms, ranging from reinforcement learning [9], [16], to the most common: genetic algorithms (GA) [17], [21].

Despite the prevalence of genetic algorithms in this domain, in our studies, we have found a simpler procedure, *Next-Ascent Stochastic-Hillclimbing (NASH)*, works as effectively as GAs and is simpler to implement and faster in practice [11]. For the training to commence, we must first specify the set of parameters that can be modified. Once specified, NASH operates as follows. A parameter is randomly chosen from the set and the modification operator for that parameter is applied. In the simplest case, if the parameter is a real number, it is perturbed by a small amount (for example $\pm 5\%$). If the parameter can take on a set of distinct values, another value is selected. Once the parameter modifications are made, the schedule is then *repaired*, if needed. The repair process ensures that the parameters are consistent with each other and are set within the appropriate ranges. For example, in the case of fixed-schedule light settings, we may want to ensure that the overall cycle of the light remains constant to keep all lights synchronized, but the individual phase lengths within the cycle can change. In this case, once a phase length perturbation has been made, the repair process ensures that the other other phase lengths are reduced appropriately to compensate and keep the overall cycle length static.

Once any repairs are made, the new schedule is evaluated with the desired *objective function*, where the objective function can be set to minimize the overall/average wait time, travel time, amount of toxic emissions, etc. If the perturbation improved the performance on the objective function over the previous settings without the perturbation, the perturbation is accepted, and the schedule with the perturbation becomes the new baseline. If the perturbation has not performed as well on the objective function, the perturbation is discarded. The previous baseline remains unchanged and the next perturbation starts from the previous baseline. For the optimization portion of this study, we set the objective function to **minimize the summed travel time of *all* the cars in the simulation**.

[1]Note that the approach presented in [6] differs from other auction-related controllers in which drivers and/or automated cars can bid for the right of way [4]. In [6], the auction serves as a unifying internal mechanism to handle the complexities of prioritizing the different phases (colors) of the lights.

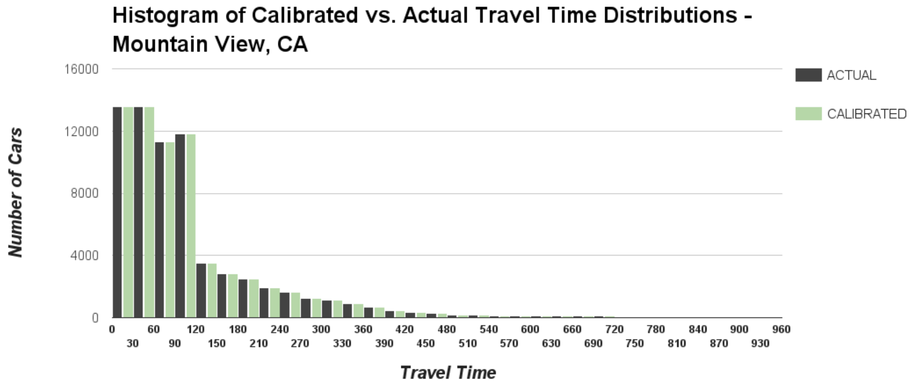

Fig. 3. Distributions of travel times with NASH-calibrated-lights and actual/observed data.

A. Training Specifics and Avoiding Overfitting

In this subsection, we briefly detail some of the training specifics for reproducibility. After the perturbation and repair process described above, the evaluation process first instantiates the controllers within the SUMO framework. Then, the recorded traffic (based on the travel tracks) is injected into the simulation and the total travel time of the cars to travel from their origin to destinations is calculated. In accordance to the objective function, the lower the total, the better the solution.

For the experiments presented in this study, a number of parameters are accessible to the training procedure for it to modify: whether a detector was used, each individual detector's weight, the minimum and maximum duration of each phase, and the light's phase offset value (this determines which phase the light starts in).

Similar to standard machine learning practice, it is important to ensure that the training process does not overfit the training data. If we simply used the travel-tracks as they were collected from the cars in Mountain View, CA, the trained controllers would reduce their travel times significantly. However, because of over-training, they would not perform well on other, even similar, scenarios. To avoid overfitting, triplets are extracted from the original travel-tracks [origin, destination, injection-time]. Here, origin and destination are geographic points where the car entered/exited the area of interest, and injection-time is the time that the car entered the zone. Recall that if the injection-time was not used in simulation, realistic traffic jams would not occur. Approximately 48,000 triplets are created. Rather than using the original set of triplets directly, prior to training, 5 related scenarios are synthesized from the original set. For each of the 5 scenarios, each triplet may be perturbed: there is a small probability of being replicated, deleted, or altering the injection time by up to two minutes. These small changes will drastically effect the overall behavior of the system. Each parameter setting's score is based on its ability to minimize the summed travel-times across all 5 scenarios. In this manner, we prevent over-training the controller to one specific scenario, while still customizing it to realistic flows.

In all of the experiments, this entire perturb-evaluate-update cycle is iterated until 2,000 candidate parameter settings are tried. The best of the 2,000 parameter setting is returned as the controller setting.

IV. Setting A Realistic Baseline

Before examining how the new sensors work, we must first have a realistic baseline performance of the deployed lights. Despite having information on where the traffic lights are and the actual traffic flows, the existing traffic light schedules and behaviors are often not accessible. Rarely are they kept in a central database, and even when they are, they are often not always obtainable. Unfortunately, without the existing schedule information, it is difficult to ascertain the real-improvements of this study — especially on traffic patterns (such as increased traffic load) that were not seen during data collection.

In the previous section, NASH was used to train the traffic light controllers to minimize travel times. In this section, we use NASH to *estimate the settings of the traffic lights that are currently installed*. Instead of optimizing the traffic lights behavior to maximize throughput, we learn the traffic light parameters which mimic the observed travel-tracks [2].

As described earlier, central to the training procedure is the *objective function*. Previously, the evaluation of the controller was based on the objective to minimize the summed time for the cars from origin to destination. Here, we specify a new objective function:

$$\min \sum_{c \in \mathcal{C}} |\text{JourneyTime}_h(c) - \text{JourneyTime}_a(c)| \qquad (1)$$

where h and a are the hypothesized and actual light settings, and $\mathcal{C}$ denotes the set of cars in the simulation. Minimizing this objective allows us to determine light settings that generate a simulated traffic flow that closely mirrors the actual flow through the same road network.

Numerous tests were conducted to verify that the timings found by using this method correspond, in behavior, to those observed in the data. Figure 3 shows the distribution of travel times obtained through simulation with the calibrated lights

TABLE I
MEAN TRAVEL TIME (MTT) UNDER INCREASING LOAD WITH MATCHED & STATIC-OPTIMIZED CONTROLLERS (UNIT:SECONDS). *With no sensors, this is independent of cell phone usage.*

	L =0.5	1.0	1.2	1.5	2.0
Baseline1: matched to existing	104	250	362	440	495
Baseline2: static-optimized	100	106	199	314	401

and those from the actual distribution. Though beyond the scope of this paper, the reader is referred to [2] for more details on matching deployed lights.

V. Experimental Results: The Right Time to Deploy Cell Phone Sensors?

In the previous section, we set the parameters on the static-light controller to match the behavior of the currently deployed lights. With those controller settings, we can test how the currently deployed lights perform under various load conditions. We scale the load, L, to inject up to $2\times$ the number of cars in the same time period. The mean travel times are shown in the first row of Table I.

As shown in Table I, when the load is increased from $0.5\times$ to $2\times$ the training load, there is the expected increase in mean travel times (MTT) — the roadways became more crowded and the traffic jams more severe. This is the first baseline; it measured how the currently deployed traffic lights would perform under increasing load.

The next row of the table establishes a much stronger baseline which is required for a more fair comparison. We continue with the *static-timing* controllers; however, the phase-timings and phase-offsets (light programs) are trained using NASH to minimize the mean travel time of the observed flows. In contrast to the first row, instead of using timings for the lights that matched those currently deployed, the *static-optimized* lights' parameters are trained to minimize travel time. Improvement is seen under all load conditions. It is important to note that still no sensors (cell phone or induction) are used. This is similar to many existing traffic lights that employ a fixed schedule that does not vary with the number of cars waiting.

With the two baselines calculated, we return to the controllers that can take advantage of sensors. The reactive controllers (as described earlier, these are based on [6]) are trained using *NASH*: in addition to the controllers' schedules being optimized, the weight given to the cell phone sensors is also optimized (details are in Section III-A). First, we examine the optimal case in which *every* driver that has a cell phone that is detected. This case also approximates the behavior of having 3 standard induction loop sensors, since all the cars are sensed. Table II (row 1) shows the results under these assumptions ($F = 100\%$). Under all loads, the performance outperformed the matched-lights and the static optimized lights. Significant out-performance was achieved when testing larger loads $L > 1.0\times$. This is expected as sensors allow the traffic lights to be adaptive to existing traffic conditions.

TABLE II
MTT (SEC.) AUCTION-BASED CONTROLLERS TESTED WITH VARYING CELL USAGE AND LOAD.

Cell Usage (%)	L =0.5	1.0	1.2	1.5	2.0
F = 100%	100	102	105	232	314
F = 50%	101	103	103	211	305
F = 20%	105	106	107	227	309
F = 10%	113	128	141	274	330
F = 2%	282	621	692	633	763
F = 0%	847	1077	1096	1113	1063

The more difficult cases arise when a smaller percentage of cars have usable cell phone signals. The second row examines what happens when the fraction, F, of usable cell phones drops by half to 50%. In this case, only 50% of the cell phone users have opted into allowing their signal to be used for traffic lights. The results are largely unchanged. Going further, an important result is witnessed, **even when F is dropped to 20%, little change in performance is seen compared to the optimal case of $F = 100\%$**. This trend does not hold when $F < 10\%$. At $F < 10\%$, there are noticeable increases across all load settings. When examining $F = 2\%$ and 0%, the controllers perform poorly in every respect, even when compared to the currently deployed traffic lights.

These results immediately raise two questions. First, why does the performance not degrade much in the beginning? This has to do with the nature of traffic. In heavy traffic times the chances of having a nearby car trigger the cell phone sensor is high. Even in less busy times, traffic is often bursty, as clumps of traffic travel in clusters through lights. Therefore, in populated regions where these intelligent lights may have the most benefit, single cars waiting at a light is rarely seen during busy travel times. The *clumpy* nature of traffic controlled by traffic lights increases the likelihood that there will be a car near you that has a cell phone that is detected.

Second, why does performance fall below the light controllers that *use no sensors*? This answer is more complex. During training, we had to select a fraction of cell phones present in the training scenarios. The results reported here are those from training with $F = 100\%$ (even though we test with a variety of settings $0 \leq F \leq 100\%$ – as shown in the table). Through training, the learning procedure found the information in the sensors useful, and thus the controller set its policies relying on readings from those sensors. In testing, we drastically changed the underlying assumptions – when F fell below 10%, the testing conditions sufficiently changed from training, and the sensors no longer provided enough information to base control decision upon. The light controllers waited, unsuccessfully, for the sensors to trigger phase changes. This caused severe gridlock.

This leads to a important question - what happens if we **train** using scenarios in which F is small? To answers this question, the full set of the experiments and training procedures was repeated multiple times with various settings for F. When $F < 5\%$ in training, the learning procedure *automatically removed all sensors and effectively converted the* *auction-based lights into static lights!* There were not enough cell phones to provide reliable inputs into the controller. When F was raised higher, but under 20%, the results were varied; some controllers used the sensors, others did not; however, they did *not* consistently outperform training with $F = 100\%$, even when tested with lower F.

Finally, it is interesting to note that when the settings for F were raised above 20% in training, the performance of the controllers performed similarly to those trained with $F = 100\%$ across all load levels in the realistic operating regions: when the fraction of cell phones in deployment increased above 10%. In all cases, below 10% actual cell phone deployment, no matter the training scenarios, the scarcity of cell phones was not overcome and either static controllers or controllers with induction sensors are preferred. In summary, when the adoption rate surpassed a relatively modest 10%-20%, it was possible - under numerous training conditions - to learn controllers that provided benefit to *all* drivers.

VI. Conclusions and Future Work

The most salient finding is that by even using only a small subset of drivers' cell phone signals, traffic light controllers can provide a benefit that is shared by all drivers. In the simulations presented here, the tipping points are between 10-20% of drivers. When that point is reached, cell phone based sensors can be deployed in heavy traffic intersections – which will likely see benefits in terms of reduced deployment cost, reduced maintenance and less disruption caused to drivers through installation and repairs. Before the tipping point is reached, however, static lights, or traditional induction loop sensors, will likely be more reliable.

The results presented in this paper are applicable to physical and virtual sensors. If the traffic lights are coordinated through a central server, it is possible to use cell phone GPS coordinates that are uploaded to a central repository to virtually measure traffic density on roads and thereby emulate (and expand upon) physical sensors. Regardless of whether the sensors are physical or virtual, many of the same adoption hurdles and concerns will need to be addressed.

There are three immediate areas for future research. If physical sensors are to be deployed, the first step is to determine the accuracy of commercial and research-based cell phone sensors for this domain. This will directly impact the percentage of users that are required to opt-in and may shift the tipping point. Second, earlier, we alluded to the problem of finding multiple cell phones in the same car and mistaking them as extra traffic. Though detecting this scenario is possible, (especially with virtual servers where the analysis is done on the server side [20]), an interesting alternative is to intentionally not distinguish this case. This would give preferential treatment to cars with more riders – thereby reducing mean wait times across passengers (not cars) and even encouraging car pooling. This is left for policy makers to decide. Finally, training with small F deserves to be revisited for further study. In particular, learning to intelligently switch between sensing and non-sensing modes may further reduce the tipping points and make this applicable to infrequently traveled roads as well.

References

[1] Asay, M.: Volkswagen: 'the car must not become a data monster'. http://readwrite.com/2014/03/11/volkswagen-data-monster-privacy-google-self-driving-cars (2014)

[2] Baluja, S., Covell, M., Sukthankar, R.: Approximating the effects of installed traffic lights: A behaviorist approach based on travel tracks. In: Intelligent Transportation Systems-2015 (2015)

[3] Barth, D.: The bright side of sitting in traffic: Crowdsourcing road congestion data. http://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html (2009)

[4] Carlino, D., Boyles, S.D., Stone, P.: Auction-based autonomous intersection management. In: 16th Intelligent Transportation Systems-(ITSC), 2013. pp. 529–534. IEEE (2013)

[5] Christensen, D., Newcomb, D.: Evaluation of induction loop installation procedures in flexible pavements. In: WA-RD 76.1. Washington State Dept. of Tranportation (1985)

[6] Covell, M., Baluja, S., Sukthankar, R.: Micro-auction based traffic light control: Responsive, local decision making. In: Intelligent Transportation Systems-2015 (2015)

[7] DLR Institute of Transportation: Sumo - simulation of urban mobility. http://goo.gl/IVPFZU (2015)

[8] D.O.T.: Extent and causes of inductive-loop failures, publication number: Fhwa-hrt-06-139 (2006)

[9] Dresner, K., Stone, P.: Traffic intersections of the future. In: AAAI-2006. pp. 1593–1596 (2006)

[10] Haklay, M., Weber, P.: OpenStreetMap: User-generated street maps. IEEE Pervasive Computing 7 (2008)

[11] Juels, A., Wattenberg, M.: Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. In: NIPS-8 (1995)

[12] Kargupta, H.: Data analytics for connected cars. http://automotive.cioreview.com/cxoinsight/data-analytics-for-connected-cars-nid-6142-cid-4.html (2015)

[13] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of SUMO – Simulation of Urban MObility. Int. J. On Advances in Sys. and Measurements 5 (2012)

[14] libelium.com website-2015: Smartphone, cellular, mobile and hand phone detection (vehicle traffic monitoring). http://www.libelium.com/products/meshlium/smartphone-detection/

[15] Lindsay, D.: Data mining the new black box of self-driving cars. http://theconversation.com/data-mining-the-new-black-box-of-selfdriving-cars-31685 (2014)

[16] Moriarty, D.E., Langley, P.: Learning cooperative lane selection strategies for highways. In: AAAI/IAAI,1998. pp. 684–691 (1998)

[17] Park, B.B., Messer, C.J., Urbanik II, T.: Enhanced genetic algorithm for signal-timing optimization of oversaturated intersections. TRB: J. Transpor. Res. Brd. 1727(1), 32–41 (2000)

[18] Priddle, A.: On-board technology in vehicles becomes latest privacy battlefield. http://archive.freep.com/article/20140116/BUSINESS03/301160043/Ford-Mulally-Fields-Farley-privacy-auto (2014)

[19] Scott, N.W.: Study of cellular phone detection techniques. University of Nebraska - Lincoln (2011)

[20] Stenneth, L., Wolfson, O., Yu, P.S., Xu, B.: Transportation mode detection using mobile phones and gis information. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 54–63. ACM (2011)

[21] Turky, A.M., Ahmad, M.S., Yusoff, M.Z.M.: The use of genetic algorithm for traffic light and pedestrian crossing control. Int. J. Comput. Sci. Netw. Sec 9(2), 88–96 (2009)

[22] Xie, X.F., Smith, S.F., Lu, L., Barlow, G.J.: Schedule-driven intersection control. Trans. Res. Part C 24 (2012)

[23] Zhang, J., Wang, F.Y., Wang, K., Lin, W.H., Xu, X., Chen, C.: Data-driven intelligent transportation systems: A survey. Intelligent Transportation Systems, IEEE Transactions on 12(4), 1624–1639 (2011)