Poster Abstract: System for Vehicle Selection in Drive-by Sensing

Dhruv Agarwal*
dhruv.agarwal_asp20@ashoka.edu.in
Ashoka University
Sonipat, Haryana, India

Srinivasan Iyengar
t-sriyen@microsoft.com
Microsoft Research India
Bangalore, Karnataka, India

Manohar Swaminathan
manohar.swaminathan@microsoft.com
Microsoft Research India
Bangalore, Karnataka, India

ABSTRACT

Drive-by sensing has emerged as a popular way to achieve fine-grained sensing of physical phenomena. However, for it to be effective at a city-scale, there is a need to optimally select a subset of vehicles from a larger available fleet. These chosen vehicles must maximize coverage of the entire city. Simultaneously, they must fulfill other deployment requirements specific to the sensing application such as reference-monitor colocation instances for gas sensors. In this paper, we describe a system to evaluate the coverage offered by different subsets of vehicles for sensor deployment based on historical vehicle mobility data. Our system allows evaluation of different vehicle selection algorithms, and also provides two in-built baselines — i) **Random-MP**, and ii) **MaxPoints** — for comparison. Finally, we provide visualizations showing coverage to gauge the efficacy of different vehicle selections.

CCS CONCEPTS

• **Computer systems organization → Sensor networks**; • **Hardware → Sensor applications and deployments**.

KEYWORDS

sensor deployment, low-cost sensing, drive-by sensing

ACM Reference Format:
Dhruv Agarwal, Srinivasan Iyengar, and Manohar Swaminathan. 2019. Poster Abstract: System for Vehicle Selection in Drive-by Sensing. In *SenSys '19: Conference on Embedded Networked Sensor Systems, November 10–13, 2019, New York, NY, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3356250.3361943

1 INTRODUCTION

Drive-by sensing is an increasingly popular way to sense physical phenomena using vehicles equipped with low-cost sensors that are driven around a geographical area of interest. This method of sensing has a number of benefits over traditional deployments. Unlike static sensors that only report value from a fixed geographic location, *drive-by sensing* provides coverage for a larger area. Thus, it requires fewer sensors than static sensing to attain the same coverage. Further, the saved cost due to fewer sensors can be used to employ more vehicles in order to increase coverage. Moreover, sensors for applications like air pollution sensing need frequent colocation with reference-grade monitors for calibration. With *drive-by sensing*, multi-hop calibration [4] techniques allow colocation to be done on the fly without suspending their operation. However, in the case of static sensors, this would require manual colocation, which in turn results in sensor downtime.

For *drive-by sensing* to be effective at a city-scale, the set of vehicles needs to be chosen carefully from a larger fleet of available vehicles, subject to an available budget. Ideally, the selected vehicles must maximize spatio-temporal coverage, while satisfying other sensing application-specific needs (such as, reference colocations).

In this paper, we present a system that provides a playground to compare sensor deployments involving different subsets of vehicles by reporting coverage obtained through each selection. Initially, the user inputs mobility patterns (GPS timestamps) of either or both cabs and public transport buses for a candidate city. The mobility dataset is split into two parts — train and test. To evaluate the performance of the user-provided vehicles, the system has two baseline vehicle selection algorithms that are run on the train set and show the coverage obtained on the test set. The system also has in-built support for the following two publicly available mobility datasets – San Francisco Taxis [8] and Rome Taxis [1]. Thus, the users can easily assess their vehicle selection algorithms against the baselines on these datasets. We intend to open-source our system for *drive-by sensing* network operators.

2 SYSTEM DESCRIPTION

In this section, we describe the implementation of our system. Further, we describe a couple of baseline comparison algorithms along with the metric **percentage coverage** used to evaluate them.

2.1 Implementation

We describe the steps implemented for evaluating the performance of the selected vehicles for a sensing application in a given city.

Step 1: Stratification - Our system provides the option of partitioning the city in three ways – (i) uniformly-sized grids, (ii) segmentation of the road network, and (iii) custom stratification polygons (for example, administrative boroughs in the city) defined in a GeoJSON-encoded polygon format. Different sensing applications may require a different kind of stratification – air pollution sensing may require uniform "gridding" of the city, pothole detection may require segmentation of the road network. Further, we also assign each resulting stratum a stratum ID for easy reference for the rest of the process. The number of strata is dependent on the spatial granularity entered by the user. Finally, the user also provides a temporal granularity and the system similarly assigns each time segment a time segment ID.

*The author contributed while in Microsoft Research India

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SenSys '19, November 10–13, 2019, New York, NY, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6950-3/19/11.
https://doi.org/10.1145/3356250.3361943

Step 2: Spatio-temporal Query Execution - In this step, the user inputs the mobility patterns representing the various vehicles available for *drive-by sensing*. For each record in the dataset, we will have a vehicle's locations (latitude and longitude) and the associated timestamps specifying their movement. We insert this dataset into a database and create a compound spatio-temporal index consisting of latitude, longitude, and timestamp. This index is formed by employing *GeoHash* technique, which divides the earth's surface into grids and encodes each location using an alphanumeric string. Indexing helps in faster execution of geospatial queries. For example, we can quickly assign each record in the data with the stratum ID and the time segment ID. Also, for some sensing applications, such as air quality monitoring using gas sensors, the geospatial indices can efficiently calculate colocation instances with reference monitors, needed for sensor calibration.

Step 3: Vehicle Selection - Our system divides the database into 2 parts (chronologically) – the train and the test set. The baseline algorithms select vehicles using the train set. Additionally, the user can input any subset of the vehicles selected using their algorithm. Finally, map-based visualizations displaying spatio-temporal coverage on the test set are shown for both the user-inputted and baseline-selected vehicles.

Real-world deployments are often dynamic with changing budgets and removal of vehicles from the original fleet. Our system handles such incremental deployment scenarios. Further, the dataset needs to have enough diversity in the cab routes for viable selection of vehicles. The user's budget must be large enough to tolerate daily variations in vehicle mobility.

2.2 Baseline Algorithms

Here, we define the two baseline algorithms for vehicle selection. In random minus minimum points algorithm (hereon referred as **Random-MP**), the required number of vehicles are chosen uniformly at random from the set of all vehicles that have reported $\geq k$ records in the dataset. In **Max Points** algorithm, the list of available vehicles is sorted in descending order on the basis of number of records they have reported. Then, the required number of vehicles are selected from the top of this list.

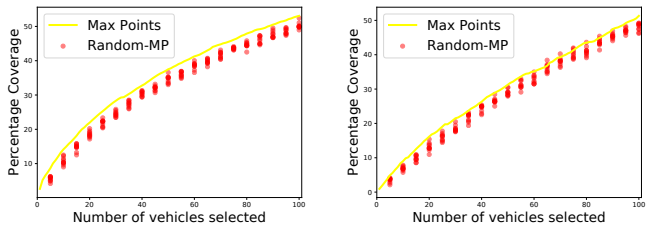
2.3 Evaluation Metric: Percentage Coverage

Let N be the set of all vehicles in a dataset that are available for deployment. Let a vehicle selection algorithm select a set M ($M \subset N$) of these vehicles. Let D be the set all spatial segments, and T be the set of all time segments. Also, let $C_{i,d,t}$ be a known binary parameter that informs if the i^{th} vehicle in the dataset was in the d^{th} ($d \in D$) spatial segment in time interval t ($t \in T$). Then,

$$\text{Percentage Coverage} = 100 \cdot \frac{\sum_{i \in M, d \in D, t \in T} C_{i,d,t}}{\sum_{j \in N, d \in D, t \in T} C_{j,d,t}}$$

3 EXPERIMENTAL RESULTS

In this section, we compare the results of using the two baseline algorithms on the datasets described earlier. For these experiments, we use the temporal granularity to be 2 hours and partition the cities into square-shaped grids of side 100 meters. We vary the overall budget for vehicle selection up to 100 for both the algorithms. For **Random-MP**, we repeat the experiment 10 times with different seeds for increasing deployment sizes in the multiple of 5. Figure 1(a) and (b) shows the results of the coverage achieved by the two algorithms on the San Francisco (SF) and Rome dataset respectively. As seen, a few cabs can cover significant portions of the two cities. For example, 40% coverage can be achieved with just 55 cabs for SF and 68 cabs for Rome through the use of **MaxPoints** algorithm. In both the cities, **Random-MP** performs worse than **MaxPoints**. However, the difference is insignificant in Rome.



(a) San Francisco (b) Rome

Figure 1: Vehicle selection: MaxPoints vs Random-MP

4 RELATED WORK

Prior work has looked at *drive-by sensing* for measuring the health of our cities. Examples include, monitoring traffic congestion [7], detecting potholes [3], sensing air quality [5], recognizing unsafe pedestrian movement [2], recording parking violations [6] etc.

5 CONCLUSION

In this paper, we presented a system to plan a *drive-by sensing* deployment in a new city. The users of our system provide data of the routes taken by a large number of vehicles (cabs, buses, etc). The system then allows the user to choose a subset of the vehicles they would like to deploy. It then compares the coverage obtained against that of the subsets chosen by our baseline algorithms.

REFERENCES

[1] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. 2014. CRAWDAD dataset roma/taxi (v. 2014-07-17). https://crawdad.org/roma/taxi/20140717. https://doi.org/10.15783/C7QC7M

[2] Trisha Datta, Shubham Jain, and Marco Gruteser. 2014. Towards city-scale smartphone sensing of potentially unsafe pedestrian movements. In *2014 IEEE 11th international conference on mobile ad hoc and sensor systems*. 663–667.

[3] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. 2008. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*. 29–39.

[4] Kaibo Fu, Wei Ren, and Wei Dong. 2017. Multihop calibration for mobile sensing: K-hop calibratability and reference sensor deployment. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. 1–9.

[5] David Hasenfratz, Olga Saukh, Christoph Walser, and Lothar Thiele. 2011. Poster: OpenSense Zurich: A System for Monitoring Air Pollution.

[6] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Chao Tian, and Yu Zheng. 2018. Detecting Vehicle Illegal Parking Events using Sharing Bikes' Trajectories.. In *KDD*. 340–349.

[7] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. 2008. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*.

[8] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD dataset epfl/mobility (v. 2009-02-24). https://crawdad.org/epfl/mobility/20090224/cab. https://doi.org/10.15783/C7J010 traceset: cab.