Modulo: Drive-by Sensing at City-scale on the Cheap

Dhruv Agarwal*
Ashoka University
dhruv.agarwal_asp20@ashoka.edu.in

Srinivasan Iyengar
Microsoft Research
t-sriyen@microsoft.com

Manohar Swaminathan
Microsoft Research
manohar.swaminathan@microsoft.com

Eash Sharma
Ola Cabs
eash.sharma@olacabs.com

Ashish Raj
Ola Cabs
ashish.raj@olacabs.com

Aadithya Hatwar
Ola Cabs
aadithya.hatwar@olacabs.com

ABSTRACT

Ambient air pollution in urban areas is a significant health hazard, with over 4.2 million deaths annually attributed to it. A crucial step in tackling these challenge is to measure air quality at a fine spatiotemporal granularity. A promising approach for several smart city projects, called *drive-by sensing*, is to leverage vehicles retrofitted with different sensors (pollution monitors, etc.) that can provide the desired spatiotemporal coverage at a fraction of the cost. However, deploying a *drive-by sensing* network at a city-scale to optimally select vehicles from a large fleet is still unexplored. In this paper, we propose Modulo – a system to bootstrap *drive-by sensing* deployment by taking into consideration a variety of aspects such as spatiotemporal coverage, budget constraints. Modulo is well-suited to satisfy unique deployment constraints such as colocations with other sensors (needed for gas and PM sensor calibration), etc. We compare Modulo with two baseline algorithms on real-world taxi and bus datasets. Modulo significantly outperforms the baselines when a fleet comprises of both taxis and fixed-route vehicles such as public transport buses. Finally, we present a real-world case study that uses Modulo to select vehicles for an air pollution sensing application.

CCS CONCEPTS

• **Information systems** → **Sensor networks**; • **Hardware** → **Sensor applications and deployments**; • **Theory of computation** → Approximation algorithms analysis.

KEYWORDS

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

ACM Reference Format:
Dhruv Agarwal, Srinivasan Iyengar, Manohar Swaminathan, Eash Sharma, Ashish Raj, and Aadithya Hatwar. 2020. Modulo: Drive-by Sensing at City-scale on the Cheap. In *ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS '20), June 15–17, 2020, , Ecuador*. ACM, New York, NY, USA, 11 pages. **https://doi.org/10.1145/3378393.3402275**

*Work done while the author was an intern at Microsoft Research India.

COMPASS '20, June 15–17, 2020, , Ecuador

ACM ISBN 978-1-4503-7129-2/20/06…$15.00
https://doi.org/10.1145/3378393.3402275

1 INTRODUCTION

Over 91% of the world's population lives in areas that exceed WHO guideline limits for air quality. Globally, an estimated 4.2 million deaths annually are attributed to ambient air pollution [2] — higher than malaria and HIV. The myriad of health complications resulting from poor air quality includes several cardiovascular and lung diseases (chronic obstructive pulmonary disease, lung cancer, asthma, etc.) [37]. One crucial step in tackling this menace is to measure the air quality at a fine spatiotemporal granularity. Access to quality data is the first step to perform wide-ranging analyses [11] that include — identifying sources of air pollution, monitoring compliance of air quality standards, measuring the efficacy of various interventions, etc. Multiple 'smart city' initiatives around the world [26] have touted to solve this problem by having a static deployment of either thousands of low-cost sensors or a few tens of reference-grade expensive monitors. Although systems to monitor our cities will certainly empower government administrators to make informed decisions, it is not without certain caveats. For example, governments around the world have installed Beta Attenuation Monitors (BAMs) for monitoring pollution levels at fixed locations around a city [9, 25]. However, installing and maintaining BAMs require major capital investment [16] — rendering them infeasible for several smaller towns. Moreover, these statically placed devices need dedicated physical infrastructure and cannot capture the spatial variations in the pollution within the monitoring region. Alternately, there are plans to install thousands of low-cost pollution monitors [3], which can generate fine-grained spatiotemporal data. However, this approach is associated with a massive administrative overhead of managing such an extensive network.

Led by improvements in mobile network connectivity, an alternate approach called *drive-by sensing* has become quite popular for monitoring large geographic areas using a fleet of vehicles equipped to sense relevant quantities. Using *drive-by sensing*, one can retrofit the vehicles with low-cost optical and electrochemical sensors to continuously monitor pollution values [21, 22]. *Drive-by sensing* leverages the mobility of vehicles to have a larger spatial footprint equivalent to multiple static monitors at a fraction of a cost. However, these low-cost sensors require continuous on-field calibration to correct for any sensor drifts. With *drive-by sensing*, vehicles can potentially rendezvous across a couple of static reference-grade monitors installed in the city a few times each day. Thus, the process of calibrating low-cost mobile sensors does not necessitate the suspension of their operation. On the contrary, static low-cost sensors would have numerous periods of downtime over their lifespan for calibration through manual colocation with expensive monitors.

Apart from air pollution monitoring, prior works have shown the efficacy of *drive-by sensing* to monitor traffic congestion [30], detect potholes [19], recognize unsafe pedestrian movement [18], record parking violations [23] and identify available parking spaces [29, 39] using in-built sensors in smartphones (IMU, cameras, etc.). Several papers have used public transportation as their choice of vehicle fleet [22]. With predictability in routes[1] and lower cost of operation, one can summon this resource for monitoring urban locations. But, many cities around the world do not have a good public transport system and, therefore, cannot guarantee sufficient coverage [17]. Fortunately, over the past few years, ride-sharing services (such as Uber, Ola, etc.) have gained tremendous popularity, and their vehicle fleets have emerged as a credible alternative. For example, as one of the largest of such ride-sharing companies, Ola is operating in >250 cities spread across multiple countries with over 1.5 million driver partners [7]. Although seemingly straightforward, piggybacking on either or both of public transport and ride-sharing services as the medium for city-scale sensing raises several complications. First, the data collection depends on the type of sensing application. Some applications may require data every few minutes, while others may require a data point once every few days. Further, the data requirement itself may not be uniform as the sensing application might need more data during a specific time and/or location. We will discuss these factors in more detail in Section 2.

Let us consider a scenario where one partners with a ride-sharing service to use their fleet for sensing air pollution in a city. The ride-sharing service may have a few thousand vehicles operating in the city, but one may have a budget to deploy only a few sensors. A natural question that arises is: *How does one select a subset of vehicles from the entire fleet that would maximize the spatiotemporal coverage of the city?* In this paper, we answer this question while accounting for the aforementioned complicating factors. Specifically, we make the following key contributions:

- **Vehicle Selection Algorithm:** We formulate the problem of selecting the optimal set of vehicles as an integer linear program. Further, we introduce several relevant extensions suitable for different *drive-by sensing* applications by changing either or both the objective and the constraints of this linear program. As the problem is **NP-Hard**, we present a greedy algorithm that gives the best-possible polynomial-time approximation algorithm for our problem.
- **System Implementation:** We introduce Modulo, our system for selecting the optimal set of vehicles for any *drive-by sensing* application. Modulo provides in-built support for using our greedy algorithm while considering various deployment-level details. Notably, for sensing applications that require calibration (gas and particulate matter sensors), Modulo uses geohash — a geospatial indexing approach — to quickly find mobile colocations with other sensors. Modulo is released as an open-source Python library for the community to utilize our approach for individual *drive-by sensing* applications. Modulo can bootstrap the process of sensor deployment in any city by examining its historical vehicular mobility patterns.
- **Detailed evaluation:** We benchmark Modulo against two baseline approaches for vehicle selection. We evaluate the performance on real-world datasets from San Francisco and Rome. Moreover, we use two kinds of datasets: one of taxis that move randomly across the city, and the other one containing buses from the public transportation system that follow fixed routes over long periods. We also conduct a case-study involving air pollution monitoring in a large Indian city using the vehicles selected by Modulo.

[1]Public transport vehicles operate on fixed routes. The routes and schedules are made available to the residents of the city to help them plan their commute.

2 BACKGROUND

In this section, we elaborate on how the spatiotemporal properties observed in the measured entity (such as pollution, potholes, traffic congestion, etc.) dictate the strategy utilized for sensing them. We also provide a brief background on the unique distinguishing characteristics of *drive-by sensing* to monitor urban environments.

2.1 Spatiotemporal properties in the sensed entity

In this paper, we introduce three complementary properties that underlies any *drive-by sensing* application — *granularity*, *variability*, and *frequency*. As elaborated below, these properties may apply to both the spatial and temporal dimensions. Understanding these properties will lead to a consequent data collection representing the characteristics exhibited by the specific sensing use-case. Next, we define these three properties and explain how they are relevant in deciding on a deployment strategy.

- **Granularity:** Granularity is the scale of data collection required for a sensing application. As discussed, granularity can be defined separately for both the space and time dimensions. Requiring high temporal granularity would translate to sensing every few seconds or minutes; while sensing at a low temporal granularity equates to recording data every few days. Similarly, high spatial granularity needs recording data every few meters; whereas, low spatial granularity means collecting data every few kilometers.
- **Variability:** Variability is the extent to which changes in the granularity of data collection are needed (or tolerated) for a sensing application. Again, variability can be defined separately for space and time dimensions. Requiring uniform spatial variability means having similar granularity in the sensed data for different regions. Correspondingly, weighted spatial variability means that the granularity of data collection changes with different areas.
- **Frequency:** Frequency is the quantity of data collected at a fixed granularity for a sensing application. We define frequency only for the time dimension. For most applications, one value sensed at a given spatiotemporal granularity might suffice. However, there could be cases where the application needs multiple sensing probes for accurate modeling of the sensed phenomena.

The spatial and temporal granularity needed to perform *drive-by sensing* vary depending on the type of applications. For example, ozone gas concentrations do not change over a few kilometers. Thus, building a *drive-by sensing* network with low spatial granularity will suffice. Detecting potholes on roads requires lower temporal

Drive by sensing application	Granularity		Variability		Frequency
	Spatial	Temporal	Spatial	Temporal	Temporal
Ozone gas	Low	Medium	Uniform	Uniform	One
Parking spot availability	High	High	Weighted towards city-centers and public areas	Weighted higher during peak times	One
Particulate matter	Medium	Medium	Uniform	Uniform	One
Pothole detection	High	Low	Uniform	Weighted; bursts in data collection tolerable	Multiple
Traffic Congestion	Medium	Medium	Uniform	Weighted higher during peak hours	One

Table 1: Drive-by sensing application needs

granularity as road conditions may not change drastically over a few days. However, we might need multiple transits from various vehicles, preferably moving on each lane to tag the road segments correctly — i.e. frequency >1 at high spatial granularity. Interestingly, some applications need both higher spatial and temporal granularity. An example application would be identifying available parking spots using cameras mounted on cabs. Nonetheless, such an application can have variability in sensing. For example, the higher granularity is necessary only during the peak hours (temporal) and at city-centers (spatial) where there is a contention on the available slots. Whereas, in the case of sensing particulate matter, we would like to have uniformity in spatial and temporal variability. Table 1 lists several applications and their associated *drive-by sensing* needs in terms of granularity, variability, and frequency.

2.2 Feasibility of Drive-by sensing

Public transport vehicles (i.e., buses, trams) or vehicles driven by working individuals with a set route and schedule have **fixed routes**. With these vehicles, there is almost complete certainty on the coverage achieved based on their historical mobility data. In cities with dense public transport network, using just the **fixed route** vehicles might suffice as the sensors placed on them can cover significant portions of the urban expanse while satisfying the properties discussed earlier. Whereas, in many cities around the world, with poor connectivity offered by the public transport system, one might have to rely on cabs operated by ride-sharing platforms. These vehicles have **random routes**, and their movement may vary significantly from one day to another. Thus, this stochasticity in their motion may lead to non-uniform coverage in different parts of the city. However, we can reduce the possible sparsity in sensing coverage with the proper selection of vehicles that cover more distance on varied routes. A composite approach would involve deploying sensors on both fleets of vehicles — i.e., those having fixed or random routes. Naturally, the choice of selecting the type of fleet will vary from city to city.

3 ALGORITHM DESIGN

We start with formulating the problem of efficient city-scale deployment of low-cost sensors on the fleet of vehicles. Given a sensing application, one must decide on the granularity and variability requirements. Based on these requirements, one can leverage vehicles willing to participate in the *drive-by sensing* exercise. Further, the optimal number of vehicles with either or both fixed and random routes can be considered using their mobility patterns. These patterns include schedules of public transport vehicles, past transit times of vehicles registered with the ride-sharing services, etc.

Let D represent the set of segments that are obtained by spatially partitioning the city under consideration for *drive-by sensing*. The segments could be line slices covering the entire road network or polygonal regions covering the whole city. Further, the spatial granularity specified decides to the cardinality of the set of segments ($|D|$). Higher the granularity, higher is the value of $|D|$. Similarly, let T represent the set of time intervals that are obtained by temporally partitioning the historical vehicular mobility data. Again, the number of the time intervals ($|T|$) depends on the minimal temporal granularity specified. Now, let us consider, n vehicles with known mobility pattern represented as a collection of sets $V = \{V_1, V_2, ..., V_n\}$. A set in this collection, represented by V_i where $i \in [1, .., n]$, contains the multiple tuples - $\langle d, t \rangle$, which represents the segment index $d \in D$ and time interval index $t \in T$. A presence of a tuple indicates that the vehicle visited segment d at time interval t. Our objective is to increase the coverage of the sensing setup containing the selected vehicles $V' \subseteq V$, such that $|V'| \leq m$. Here, m is the maximum number of vehicles that can be summoned ($m \leq n$). Thus, we want to maximize the union of the set containing $\langle d, t \rangle$ tuples visited by the selected vehicles -

$$\left| \bigcup_{V_i \in V'} V_i \right|$$

3.1 Integer Linear Programming Formulation

Here, we present a linear programming formulation that maximizes the term described earlier. Let x_i be a binary decision variable representing if the $i \in [1, ..., n]$ vehicle is selected by our optimization. Further, let m be the maximum budget of vehicles that can be selected for a given *drive-by sensing exercise*. Let $O_{d,t}$ represent the occupancy obtained from these selected vehicles at time interval $t \in T$, i.e., the set of all time intervals, and $d \in D$, the set of segments. This variable again takes binary values. $O_{d,t} = 1$, if the segment d at time interval t was occupied by any of the selected vehicle. Let $C_{i,d,t}$ represent a known binary parameter that describes if the i^{th} vehicle travelled to segment d at time interval t. Based on these parameters and variables, we define our integer linear programming formulation as:

$$\max \quad \sum_{d \in D,\, t \in T} O_{d,t} \tag{1}$$

$$\text{subject to:} \quad \sum_{i \in [1,\ldots,n]} x_i \leq m \tag{2}$$

$$\sum_{i \in [1,\ldots,n]} (C_{i,d,t} \cdot x_i) \geq O_{d,t} \quad \forall d \in D, t \in T \tag{3}$$

$$O_{d,t} \in \{0,1\} \quad \forall d \in D, t \in T \tag{4}$$

$$x_i \in \{0,1\} \quad \forall i \in [1,\ldots,n] \tag{5}$$

We expand on the choice of the objective function and the constraints mentioned above as follows:-

- **Objective function:** We want the selected vehicles to travel to as many segments at different time intervals to ensure maximum coverage. Thus, we define our objective function shown in (1) to maximize the sum of the binary variable occupancy $O_{d,t}$ over every $d \in D$ and $t \in T$.
- **Budget constraint:** Out of all the vehicles for which the mobility patterns are known, we want to select a subset of them based on a predefined budget. Hence, the sum over the binary variable x_i, representing selected vehicles, cannot cross the budget allocated (m), as shown in (2).
- **Coverage-Occupancy constraint:** If the occupancy variable $O_{d,t} > 0$, then at least one of the selected vehicles was present in segment d at time interval t (see (3)).
- **Occupancy binary constraint:** As defined earlier, occupancy $O_{d,t}$ is a binary variable. This constraint (see (4)) ensures that the variable can only take one of the two values of 0 and 1 — i.e., multiple transits from the selected vehicles are not counted multiple times.
- **Vehicle binary constraint:** As defined earlier, the variable x_i is a binary variable. This constraint (see (5)) ensures that the decision variable on the selection of vehicles can only take one of the two values of 0 and 1.

3.1.1 Algorithm Analysis. The above formulation exactly matches the classical *maximum coverage problem* — a widely studied problem in the theoretical computer science and operations research community. Unfortunately, the *maximum coverage problem* is **NP-hard** [24]. Thus, our problem cannot be solved exactly in polynomial time, if **P** ≠ **NP**.

3.1.2 Calibration considerations. The described integer programming formulation provides a general framework for selecting vehicles for *drive-by sensing*. However, a few sensing applications require placing additional constraints. For example, in the case of city-scale deployment of low-cost gas or particulate matter sensors, one needs to ensure that they are calibrated on-field regularly. This obligation necessitates the low-cost sensors to rendezvous around reference-grade pollution measuring stations [2] or other low-cost sensors that have been recently calibrated. To guarantee calibration, we can easily extend the above formulation by ensuring that the selected sensors have mobile colocations with either or both of reference-grade and other low-cost sensors. Let us consider the parameters Lb_i and Ls_i represent the number of colocations of the i^{th} vehicle with reference-grade and other low-cost sensors respectively. Then we can add the following constraints -

$$Lb_i \cdot x_i \geq b \quad \forall i \in [1,\ldots,n] \qquad \text{(Reference colocations)},$$
$$Ls_i \cdot x_i \geq s \quad \forall i \in [1,\ldots,n] \qquad \text{(Low-cost sensor colocations)}$$

Above, b and s represent the minimum number of colocations needed for selected vehicles with reference-grade and other low-cost sensors, respectively.

3.1.3 Budget Minimization Formulation. We can easily change the above formulation from maximizing coverage for a given budget to minimizing the budget necessary to meet a specific amount of coverage. For this modification we can replace the objective functions and the budget constraint, while keeping the other constraints as is in the following way:

$$\min \quad \sum_{i \in [1,\ldots,n]} x_i \qquad \text{(Budget minimization)}$$
$$\text{subject to:} \quad \sum_{d \in D,\, t \in T} O_{d,t} \geq k \qquad \text{(Minimum coverage)}$$

3.1.4 Weighted Coverage Formulation. As described in section 2, some sensing applications might require the variability in coverage to be weighted at certain spatial and temporal levels (see the "Parking Spot Availability" sensing application in Table 1). The above integer linear programming formulation can easily accommodate this by modifying the objective function in the following way:

$$\max \quad \sum_{d \in D,\, t \in T} O_{d,t} \cdot W_{d,t} \qquad \text{(Weighted objective function)}$$

Here, $W_{d,t}$ represents the weights given to the segment $d \in D$ at time interval $t \in T$. For uniform variability, the $W_{d,t} = 1\ \forall d \in D$ and $t \in T$. The weighted maximum coverage problem is also widely studied in the literature [31].

3.2 Greedy Approximation

As discussed earlier, the maximum coverage problem is **NP-hard**. Fortunately, there exists a greedy heuristic that provides a solution within an approximation ratio of $1 - \frac{1}{e}$, i.e., the best polynomial-time approximation algorithm — unless **P** = **NP** — for both unweighted and weighted maximum coverage problem [24, 31].

We modify the same greedy algorithm (see Algorithm 1) to include additional constraints such as to ensure a minimum number of reference and low-cost sensor colocations. This algorithm computes a list called *vehicles* that maximizes the weighted coverage. Similar to the notations used earlier, we have a predefined budget m, a binary parameter $C_{i,d,t}$, which indicates if the i^{th} vehicle was in segment d at time interval t. $O_{d,t}$ is a binary variable that indicates if the segment d at time interval t was covered by the selected vehicles. Functions $refColocs(j)$ and $sensorColocs(j)$ return the number of colocations of the j^{th} vehicle with reference-grade monitors and other low-cost sensors, respectively.

[2] Pollution measurement through Beta Attenuation Monitors – a gravimetric method

Algorithm 1 Greedy Sensor Deployment for Drive-by Sensing

Parameters: i) $C_{i,d,t} \forall i \in [1,...,n],\ d \in D,\ t \in T$; ii) $minRefColocs$; iii) $minSenColocs$ iv) m
Initialize: i) $currSum = 0$; ii) $vehicles = []$; iii) $O_{d,t} = 0 \forall d \in D \quad t \in T$
for i in [1,...,m] **do**
  for j in [1,...,n] **do**
    if $refColocs(j) >= minRefColocs$ **then**
      if $sensorColocs(j) >= minSenColocs$ **then**
        $\dot{O}_{d,t} = O_{d,t} \cup C_{j,d,t} \quad \forall d \in D \quad t \in T$
        if $currSum < \sum_{d \in D, t \in T} \dot{O}_{d,t} \cdot W_{d,t}$ **then**
          $selectVehicleIndex = j$
          $currSum = \sum_{d \in D, t \in T} \dot{O}_{d,t} \cdot W_{d,t}$
  $O_{d,t} = O_{d,t} \cup C_{selectVehiclesIndex,d,t} \quad \forall d \in D \quad t \in T$
  $vehicles.append(selectVehiclesIndex)$
return $vehicles$

3.2.1 Handling dynamism in deployments. In any *drive-by sensing* exercise, there might be cases where one would have to swap the sensors from one vehicle to the other for several reasons. For example, there could be churn in the driver pool of a ride-sharing service or a few participating cab drivers may not want to continue with the sensing application. Further, with an increased budget, there is scope for adding new vehicles to the sensing fleet. For long-term deployments of *drive-by sensing*, such dynamism would be commonplace. Thus, it is essential to support cases where one would like to have a new list of vehicles for an incremental deployment. For this, we can modify the existing greedy algorithm by changing the parameters and initializations to reflect the incremental case. In parameters, $C_{i,d,t}$ will be a binary parameter showing remaining vehicles that are not part of the sensing application, and m will be the remaining or additional budget available. Likewise, in initializations, $vehicles$ will contain the current set of vehicles and $O_{d,t}$ will have the occupancy from these vehicles. Whereas, $currSum$ is set to $\sum_{d \in D, t \in T} O_{d,t} \cdot W_{d,t}$, calculated using the current vehicles.

4 MODULO: SYSTEM IMPLEMENTATION

In this section, we describe Modulo, our system to identify the optimal set of vehicles that must be chosen by operators of large-scale *drive-by sensing* networks (from hereon referred just as operators). Modulo is designed to be *application-agnostic*, i.e., it is general enough to serve a plethora of sensing use-cases. Figure 1 shows the complete pipeline of Modulo consisting of the three key steps.

4.1 Step 1 - Stratification

In this step, the operator inputs a list of coordinates (latitude and longitude) encompassing the chosen geographical area to partition into smaller regions. Further, the operator provides the spatial granularity required by the sensing application (as described in Section 2). The operator also provides a division-type and weights for the stratification. They can choose one of the two in-built types of strata — i) Square-shaped grids, or ii) Road segments. This feature is provided as different sensing use-cases may require different kinds of stratification of the city. For air pollution sensing, it could be uniformly sized grids. Whereas, for pothole detection, it could be segments of the road network. The resulting strata are outputted as a GeoJSON file. Alternatively, the operator can provide a custom GeoJSON file with partitioning of the geographical region as GeoJSON encoded polygons. Finally, Modulo assigns each stratum (each grid, road-segment, or custom stratum) a unique stratum ID, which gets embedded in the exported GeoJSON file.

4.2 Step 2 - Spatiotemporal Query Execution

In this step, the operator inputs historical mobility data of the fleet that they want to use for deployment. The operator also input a temporal granularity that is required for their use case. The spatial and temporal granularity together decide the coverage of the deployment. Additionally, the administrator may also input the locations and time frequencies of any reference-grade sensors they may have deployed across the city. This is important for use-cases like air pollution sensing, where the low-cost mobile sensors need to be calibrated using the colocated reference-grade sensors. Modulo processes the historical data provided in three stages:

- The raw data is parsed, and for each record, the timestamp, GPS location, and vehicle ID are inserted into a NoSQL MongoDB [4] database. The mapping between the *personally identifiable information* (PII) and assigned vehicle IDs is encrypted and made available only to network administrators. Thus, any PII like vehicle number, driver names, etc. are stripped from our main database.
- A compound spatiotemporal index is created on this database for efficient querying of data. We choose MongoDB as our database because it implements a geospatial index by calculating a geohash [3] on the locations of the records.
- Using the geospatial index and the GeoJSON resulting from step 1, we efficiently assign a `stratum_id` for each record depending on the stratum that they fall under. The records are also assigned a `time_interval_id` depending on the temporal granularity inputted by the operator.

After these three stages, every NoSQL record in the database has the following structure:

```
vehicle_id: <int>,
stratum_id: <int>,
timestamp: <int|Unix timestamp>,
time_interval_id: <int|Unix timestamp>,
location: {
    "type": "Point",
    "coordinates": [<float|lng>, <float|lat>]
    }
```

With this setup in place, Modulo runs queries to find the coverage achieved by each vehicle in the dataset as per the `stratum_id`s and `time_interval_id`s assigned to them. Further, if needed, Modulo provides support to run queries to find colocations in the mobility data. Finally, this database setup is made available to the next stage in the pipeline to improve upon the achieved coverage.

[3] A geohash divides the earth's surface into grids and encodes a location into an alphanumeric string. These string have similar prefixes for places near to each other [32].

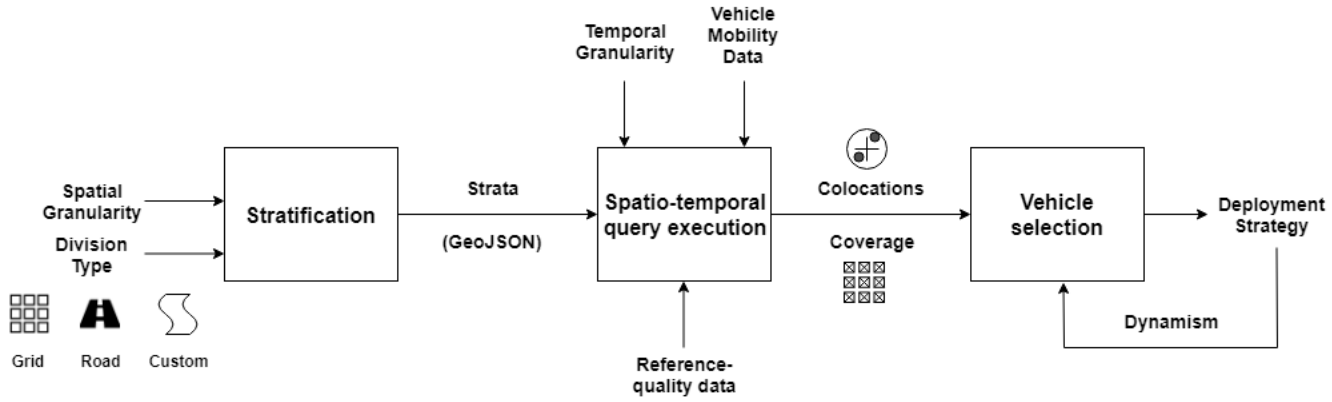


Figure 1: Pipeline highlighting the three steps involved in Modulo

(a) Grids in Rome (b) Roads in Rome (c) Wards in Blr.

Figure 2: We show the different division types enabled by Modulo. (c) shows the division of Bangalore on the basis of administrative boundaries defined in a GeoJSON format. Similarly, any other arbitrary boundaries suiting the sensing application can be handled by Modulo.

4.3 Step 3 - Vehicle Selection

In the final step of the pipeline, Modulo runs a greedy approximation solution to the Maximum Coverage Problem as detailed in Section 3.2. The result of this algorithm is a deployment strategy for the network. More specifically, the greedy algorithm returns a **set of vehicles** that should be used for deployment of the drive-by sensing network for improved coverage. This final step of the pipeline can be run every few days by the operator in order to account for dynamism in the sensor network deployments.

4.4 Open-source Python Library

We release Modulo as an open-source Python Library [4]. This library exposes APIs to perform the different steps in the Modulo pipeline. We have explained the specifics of input formats (such as JSON specification for vehicle mobility data, etc.) in the documentation accompanying the library. Further, the library repository also contains easy-to-follow examples in the form of iPython notebooks.

5 EVALUATION METHODOLOGY

In this section, we provide a comprehensive description of the datasets used in our evaluation. Further, we present two non-trivial baselines for selecting vehicles from their past mobility data. Next, we will describe the experimental setup in detail. Finally, we introduce the metric used to evaluate the efficacy of Modulo.

5.1 Dataset description

For real-world evaluation, we used taxi-tracking datasets for the city of San Francisco [36] and Rome [12] from CRAWDAD. We also found public transport transit data for the buses operated by the San Francisco Municipal Transportation Agency [5]. These two cities differ from each other in three factors that affect transportation:

- Geography – The two cities are in different continents (San Francisco in N. America and Rome in Europe)
- Population density – Rome: 2,232/km^2 [38]; San Francisco: 7,272/km^2 [13]
- City extent – Rome: 1,285 km^2; San Fran.: 600.59 km^2 [14]

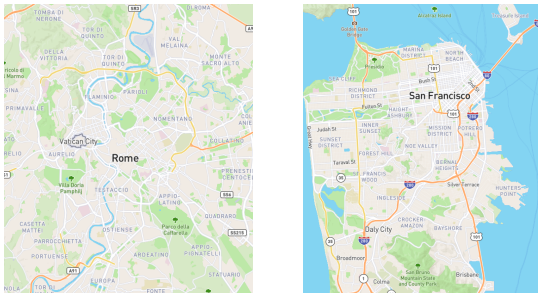
The San Francisco taxi dataset contains 11,219,878 records over 25 days in 2008 and reports GPS data every 1 minute. The Rome taxi dataset contains 21,817,850 records over 25 days in 2014, but it reports GPS data every 15 seconds. Hence we consider one in every four records to get minute-level data to maintain consistency across experiments. Each record in both the datasets contains a timestamp, taxi ID, and the corresponding GPS location. In both these datasets, we perform our experiments over seven days. The San Francisco bus dataset contained 5,610,179 records over 7 days in 2013. For realistic comparison with the San Francisco taxi dataset, we obtain records on the same month and dates as those of the taxi dataset and transpose them from 2013 to 2008.

	Records	Vehicles	Time Period
San Francisco (Bus)	5603166	627	7 days
San Francisco (Cabs)	2977508	511	7 days
Rome (Cabs)	631535	304	7 days

Table 2: Summary of datasets used

For our experiments, we consider only the records in a rectangle around the center of the city. We do this to limit the number of grids we divide the cities into (as explained in section 4) for computational convenience. The selected rectangle covers about 90%

[4] `https://github.com/microsoft/Modulo`



(a) Selected area of Rome (b) Selected area of San Fran.

Figure 3: Selected areas of San Francisco and Rome for our experiments. In [minimum longitude, minimum latitude, maximum longitude, minimum longitude] format, the bounds for Rome are: [12.4, 41.825, 12.575, 41.975]. The bounds for San Francisco are: [-122.35, 37.67, -122.515, 37.83]. Courtesy: Maps plotted using Mapbox.

of the records in both San Francisco and Rome. In terms of area, this rectangle covers 250 km^2 of San Francisco, and 240 km^2 of Rome. In San Francisco, a substantial part of this rectangle covers oceans (for example, the San Francisco Bay) to be able to cover the Golden Gate Bridge and Treasure Island. After selecting the records inside the rectangle over seven days, the dataset that we use for our experiments is summarized in table 2.

In addition to the taxi data, we also consider data from reference-grade air pollution sensors in these two cities. Since we do not know of the existence of the reference-grade sensors in 2008, we use the current (August 2019) locations of these sensors from AQICN.org [6]. Further, we assume that each of these sensors reports data at a 15-minute interval [5]. Here, as well, we only consider the reference-grade sensors located inside our rectangle of interest. According to AQICN.org, there is just one reference-grade monitor in San Francisco and four in Rome. We consider this data in addition to the vehicle data to investigate the number of "colocation" or "rendezvous" instances of taxis with high-grade air pollution sensors for highly accurate 1-hop calibration [20].

5.2 Baseline Description

In this section, we describe the two baseline methods against which we compare the results of our algorithm.

- **O'Keeffe et al.** – In the approach used by O'Keeffe et al. [35], the required number of cabs are chosen randomly from the set of vehicles.
- **Maximum Points** – In this method, the vehicles reporting the maximum number of records in the vehicle mobility dataset are selected [8] without considering other factors.

5.3 Experimental Setup

In this section, we explain the setup we use for performing our experiments. While using Modulo, we chose 'grids' as the division type. Further, we input the temporal granularity to be 2 hours and the spatial to be 100 meters. For a fair comparison among the three methods (Modulo + 2 baselines), we divide the seven days of the taxi and bus data into two halves of equal periods. This splitting is akin to the training and test set in case of evaluating machine learning algorithms. The data corresponding to the first period was used by the three methods to compute the list of selected vehicles. We evaluated the three methods by using the vehicles selected on the data corresponding to the second period.

[5]Reference monitors typically report one value every 15 minutes.

5.4 Evaluation Metric: Percentage Coverage

We design an intuitive metric to compare the effectiveness of Modulo with the other baselines. Let N be the set of all vehicles present in a dataset and let a techniques select M vehicles. Similar to the notations used in section 3, we have $C_{i,d,t}$, i.e., the binary variable representing if the i^{th} vehicle was in the d^{th} segment at time interval t. $W_{d,t}$ represents the weight of the d^{th} segment at time interval t. We define the metric **Percentage Coverage**, as follows:

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

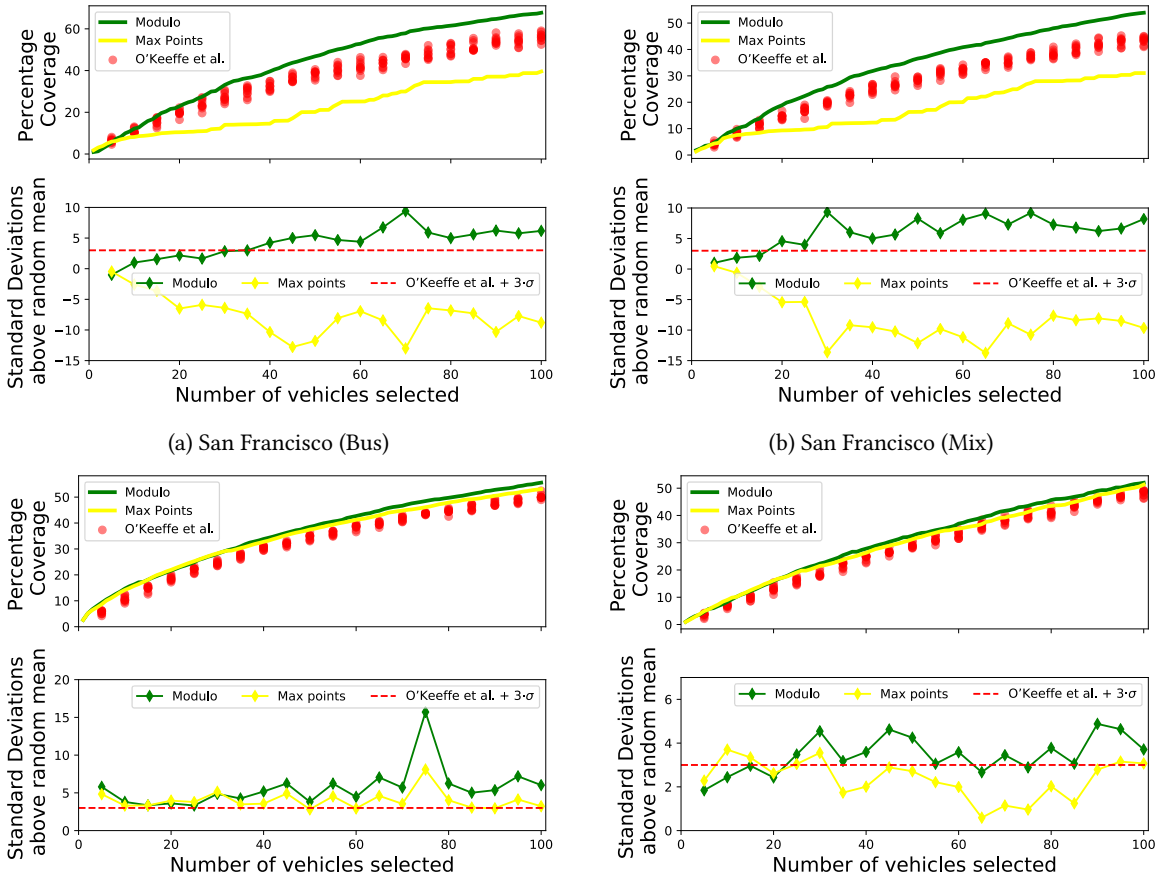
6 EXPERIMENTAL RESULTS

In evaluation, we seek to gain insights on the following questions – (i) How effective is Modulo in selecting vehicles on different datasets containing different vehicle types?, (ii) How effective is Modulo in running spatiotemporal queries?

6.1 Baseline Comparisons

We examine the performance of our approach, Modulo, as compared to the performance of the two baselines, O'Keeffe et al., and Max Points. We first divide our dataset into two halves of 3.5 days each. We then allow each of the three approaches to select vehicles from the first set that must be deployed in the second set for increased coverage. Finally, we calculate the performance metric of **percentage coverage** obtained by the selected vehicles from each approach in the second set. We repeat the experiment for a selection budget of up to 100 vehicles. In case of O'Keeffe et al., we runs 10 experiments with different seeds for different deployment sizes in the multiple of 5. Figure 4 shows the results of this experiment on 4 datasets: San Francisco bus dataset, San Francisco mixed dataset (cabs + buses), San Francisco cabs dataset, and Rome cabs dataset.

Figure 4 (a) shows the difference in the performance of the three approaches for the San Francisco bus dataset. The plot on the top indicates that for fewer buses, all the three approaches provide similar percentage coverage. However, as the deployment size increases, the percentage coverage resulting from Modulo starts to overtake the performance of O'Keeffe et al. and Max Points. Specifically, we note that to achieve 40% coverage, Modulo requires 39 buses, O'Keeffe et al. requires 55 buses ($\approx$ 41% extra), and Max Points requires a whopping 92 buses ($\approx$ 136% extra). The plot on the bottom shows the variation in performance between Modulo (green line) and Max Points (yellow line) with the mean of the 10 experiments of O'Keeffe et al. As seen, for a deployment size of more than 30 buses, Modulo always performs more than 3 standard deviations (indicated as a dotted red line) better than the mean of the O'Keeffe



(a) San Francisco (Bus)

(b) San Francisco (Mix)

(c) San Francisco (Cabs)

(d) Rome (Cabs)

Figure 4: Coverage through drive-by sensing for San Francisco and Rome (Modulo vs Max Points vs O'Keeffe et al.)

et al. deployment. This behavior is explained by the fact that buses ply on fixed routes, which results in the predictability of routes in the future. Since Modulo uses historical trends to select cabs, this predictability allows it to provide increased coverage in the future as well. Figure 4 (b) shows similar results when the three approaches are applied to the dataset containing both fixed-route vehicles (buses) and random-route vehicles (taxis). Modulo performs better than O'Keeffe et al. and Max Points, and offers more than 3 standard deviations higher coverage that the O'Keeffe et al. baseline. However, in this case, O'Keeffe et al. performs much worse than in figure 4 (a) with respect to the performance of Modulo. This is expected since O'Keeffe et al. now has a mixed set of vehicles to pick. Hence, it may pick a few cabs, as opposed to picking all buses with predictable routes with higher coverage in figure 4 (a). Surprisingly, Max Points picked up just 2 cabs in its selection of 100 vehicles, whereas Modulo picked as many as 37 cabs out of the overall 100. This suggests that although buses have predictable routes, they might be spending a lot of time in similar locations and could have sparse coverage. A common trend observed in both Figure 4 (a) and Figure 4 (b) is that there is diminishing return when the deployment size increases.

Observation. *Modulo outperforms both O'Keeffe et al. and Max Points when fixed route vehicles are allowed to be picked because they allow applicability of past trends to the future. However, even in case of mixed routes, Modulo picks vehicles based on their past coverage and thus performs significantly better. As the vehicle count increases, there are diminishing returns in percentage coverage, i.e., the increase in the deployment size follows a sub-linear trend.*

Figure 4 (c) and (d) show the performance results of the three approaches in datasets comprising only of taxis, i.e., San Francisco and Rome, respectively. In this scenario, there is only 2-5% increase in the percentage coverage offered by Modulo over Max Points, and 7-11% increase over O'Keeffe et al., when the size of deployment is higher than 40%. This difference in performance is lower for smaller deployments. This result suggests that the historical movement data of taxis is not indicative of the trends they would follow in the future. Neither of the three approaches benefits from any predictability in the movement patterns. In San Francisco, Modulo and Max Points approaches are consistently more than three standard deviations better than the mean values for O'Keeffe et al.. However, in Rome, Max Points struggles to offer significant improvement, i.e., less than two standard deviations, above the mean.

Observation. *Lack of predictability in taxi movement results in no significant outperforming of one approach over the other. In these cases, the absolute performance of Modulo over other methods is marginal; there is still a significant statistical difference in performance.*

6.2 Modulo Runtime Results

In these experiments, we show the efficiency of Modulo in performing spatiotemporal queries on a large database. We perform these experiments on the original databases without removing any records. However, the process of retrieving per-minute values instead of per-15-second values results in the reduction of the Rome dataset from 21,817,850 records to 3,094,358 records. Hence, we perform this experiment on the San Francisco database of size 11,219,878 records and Rome database of size 3,094,358 records.

For the experiment shown in figure 5, we choose 500 random coordinates associated with records in the San Francisco and Rome databases. We run a spatiotemporal query to find all colocations around these coordinates, i.e., all points in its 50 meters spatial radius and 5 minute temporal radius. The reported time is the total time required by the database for query plan selection and query execution in milliseconds [4]. The mean query time for the Rome database is 75.55 milliseconds and for the San Francisco database in 99.7 milliseconds. This difference is expected as the San Francisco database is almost 4× as large as the Rome database.

Observation. *The average query time in the Rome and the San Francisco database is 75.55 and 99.7 milliseconds, respectively. There is a correlation between query time and the database size.*

Figure 6 and figure 7 show another experiment on Modulo's geospatial database. In this experiment, we test the performance of the database on arbitrarily sized polygons. Efficient resolving of these queries is important for Modulo to be able to provide in-built support for custom stratification. Further, this allows data stored in Modulo database to be ideally suited for social science experiments involving interventions created in different portions of a given city. To evaluate the performance of Modulo for arbitrarily sized regions, we use a GeoJSON file of polygons defining the neighbourhood boundaries of San Francisco [1]. Then, for each neighborhood, we query our database for all the records lying in it. We report the time taken for each such query with respect to two metrics: the area of the neighborhood in figure 6, and the number of points returned in the neighborhood 7. The Pearson's r coefficient for the area-time plot is 0.363 and for the number-time plot is 0.797.

Observation. *There seems to be a weak correlation between the area of an arbitrary polygon and the query time. However, there appears to be a stronger correlation between the number of points in a polygon and the query time.*

7 MODULO: REAL-WORLD DEPLOYMENT

India's National Capital Region, which consists of Delhi and neighboring districts in the surrounding states, suffers from very high pollution levels. Specifically, in the winter months, the PM2.5 levels are higher than 300 $\mu g\,m^{-3}$ [33] — an order of magnitude above the WHO prescribed guidelines of 10 $\mu g\,m^{-3}$ [34]. Thus, we wanted to deploy our *drive-by-sensing* approach using Modulo in this region. For this purpose, we chose vehicles leased by driver-partners from OLA-owned fleet. Initially, we prefiltered vehicles based on their age (<3 years old) and the number of days the driver has driven for OLA (>300 days). These prefiltered cabs from the overall fleet of the ride-sharing service were given as input to Modulo. For this case study, we chose a spatial granularity defined by GeoHash-5 and a temporal granularity of 1 hour. We chose a uniform spatial and temporal variability, and a temporal frequency of one. We looked at the mobility data of prefiltered cabs for a period of one month in October 2019. For comparison, we also used O'Keeffe et al.'s approach to select cabs from the mobility data of the available cabs.

The results for this deployment over a 7 day period are shown in Figure 8(a). Irrespective of the number of cabs selected (i.e., the budget), Modulo consistently outperforms O'Keeffe et al. Further, as deployment size increases, the quantum of improvement that Modulo offers over O'Keeffe et al. also increases. Due to budget limitations, we retrofitted around 20 cabs with the optical PM2.5 sensor (Figure 8(b)). We have been regularly monitoring the pollution values since December 15, 2019 and intend to release the dataset once every quarter starting June 2020. A snapshot of our coverage is shown in Figure 8(c).

8 RELATED WORK

8.1 Urban drive-by sensing deployments

Several papers have looked at *drive-by sensing* as an effective method for collecting data for various use-cases [18, 19, 21–23, 29, 30, 39]. The applications provide a promising overview of how *drive-by sensing* can be used in different urban environments. Although these are exciting applications, a detailed study on the city-scale viability of their approach using *drive-by sensing* is not discussed. Anjomshoaa et al. provided an empirical evaluation on the street coverage obtained from fixed-route and random-route vehicles. However, beyond a few statistical results, this work does not present a complete strategy on selecting an optimal set of vehicles based on their study. Ali et al. [10] looked at the application of pothole detection and modeled the sensor deployment as a maximum coverage problem. However, they only looked at vehicles having predefined routes and do not consider the extensions needed for serving several use-cases for all kinds of *drive-by sensing* application. Liu et al. presented a detailed survey on a broader topic of mobile crowdsensing [27]. They also talk about several works using *drive-by sensing*. But the sensor selection problem for *drive-by sensing* is not discussed. A recent work by O'Keeffe et al. [35] show interesting statistical results on cabs registered on ride-sharing and city taxis. However, they do not go beyond the random selection of taxis.

8.2 Air Quality sensing

An exciting use-case of *drive-by sensing* is for air quality monitoring. This application has several unique challenges, such as on-field calibration of the gas and PM sensors, localized phenomena of pollution, etc. Several papers have utilized sensors deployed on moving vehicles to collect pollution data [20, 21]. Air quality monitoring is a perfect application for *drive-by sensing* as pollution is a local phenomenon, and there are significant variations from one place in the city to the other. Interestingly, Fu et al. have looked at the optimal placement of expensive monitors to make low-cost sensors k-hop calibrable [20]. They formulate the problem as a set cover problem. However, these results on k-hop calibrability apply only to

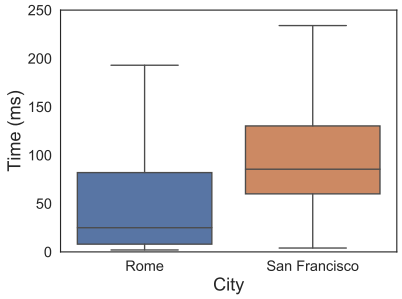


Figure 5: Colocation query time

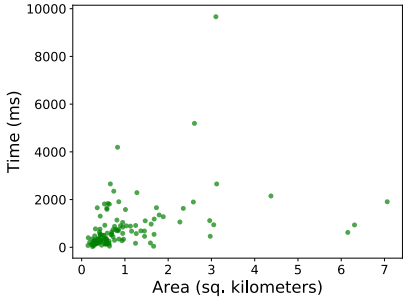


Figure 6: Query time vs Area

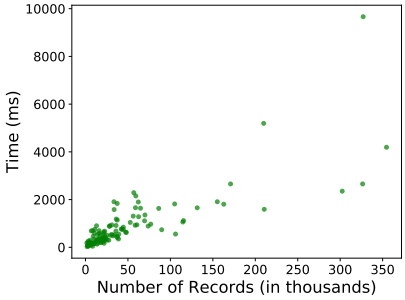
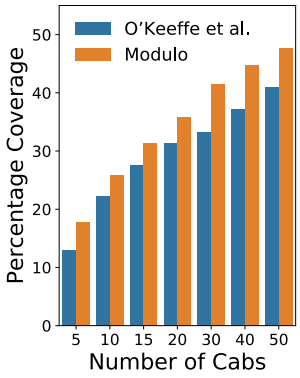
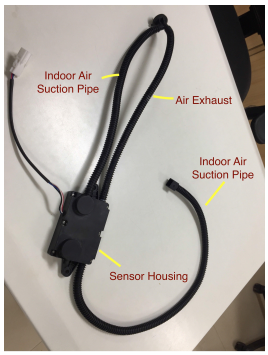


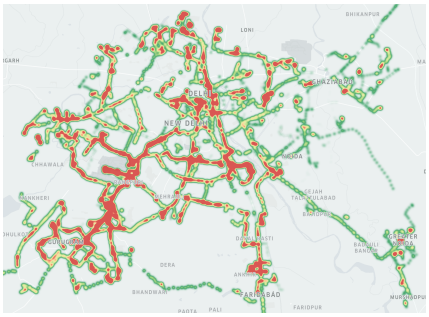
Figure 7: Query time vs Points found



(a) Percentage Coverage for NCR



(b) PM Sensor Deployed



(c) Coverage obtained for NCR on Jan 3, 2020

Figure 8: Case study with OLA-owned fleet in Delhi/NCR

fixed routes vehicles. Saukh et al. also looked at the route selection problem to achieve coverage for fixed-route vehicles [40].

8.3 Feasibility of drive-by sensing

Drive-by sensing approach necessitates the use of low-cost sensors that raises a few questions on the reliability and accuracy of the data generated. Recent works have shown that maintaining sensor accuracy is easier through *drive-by sensng* [20, 28] using on-field mobile calibration. Further, a recent work explores the idea of sensor reliability at a large scale using a sensor's output voltage-based primitive called the "Fall-curve" [15]. However, there are privacy concerns when using *drive-by sensing* data. In our case, we ensure that the vehicle data is anonymized. However, one can follow the approach presented by Hoh et al. to evaluate and ensure privacy in GPS traces while maintaining sufficient data accuracy.

9 CONCLUSION AND FUTURE WORK

Drive-by sensing is an upcoming way of sensing the physical phenomena around us with fine-granularity at a city. But for city-scale deployments to be successful, the vehicle fleet used for the deployment needs to be carefully selected so that it fulfils all the requirements of the sensing application like coverage at a specified spatial and temporal granularity, colocations for calibration, dynamism in the deployment, etc. In this paper, we proposed Modulo – a novel approach to the deployment of vehicles for **drive-by sensing** that is generalize to multiple sensing applications. We expose a Python library that enables sensor network operators to use this approach for selecting the optimal set of vehicles from a fleet of candidate vehicles. We also propose variations of our algorithm to suit specific needs like budgeted deployment and weighted coverage. We compare our approach against a couple of baseline algorithms on three kinds of datasets: all cabs, all buses, a mix of cabs and buses. We see that our algorithm outperforms the other baseline methods. Specifically, for a *drive-by sensing* application in the city of San Francisco, we obtained 40% coverage using just 39 public transport buses. We also conducted a real-world case study where we used Modulo to select vehicles for an air pollution sensing application.

As part of our experiments and the real-world deployment, we observed that specific locations in a city could be underserved or inaccessible (large parks, etc.) by almost all the transportation services (both public and ride-sharing services). Thus, we intend to incorporate features in Modulo to allow a budget for static monitors for opportunistically placing them in such parts of the city. Such a feature would allow sensor network operators to study the impact of allocating different budgets for static and mobile monitors on the overall coverage. Moreover, any offline vehicle selection algorithm, such as Modulo, cannot guarantee real-time coverage. As part of our future work, we plan to study the role of incentives for drivers to take minor detours to improve the overall spatial coverage to cover spatiotemporal holes in sensing. A well-designed incentive scheme can empower Gig workers, such as food delivery agents and driver of ride-sharing vehicles, to earn extra income in the post-COVID-19 world.

REFERENCES

[1] 2016. SF Find Neighborhoods. (visited on August 2019). **https://data.sfgov.org/Geographic-Locations-and-Boundaries/SF-Find-Neighborhoods/pty2-tcw4**. (2016).

[2] 2019. Air Pollution (visited on March 2020). **http://www9.who.int/airpollution/en/**. (2019).

[3] 2019. Mayor launches world's largest air quality monitoring network. **https://www.london.gov.uk/press-releases/mayoral/to-identify-londons-toxic-air-hotspots**. (2019).

[4] 2019. MongoDB. (visited on August 2019). **https://www.mongodb.com/**. (2019).

[5] 2019. SFMTA Transit Data. (visited on August 2019). **ftp://avl-data.sfmta.com/avl_data/avl_raw**. (2019).

[6] 2019. World Air Quality Index. (visited on August 2019). **http://aqicn.org/**. (2019).

[7] 2020. About Ola. **https://www.olacabs.com/about.html**. (2020). [Online; accessed 02-March-2020].

[8] Dhruv Agarwal, Srinivasan Iyengar, and Manohar Swaminathan. 2019. System for vehicle selection in drive-by sensing. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 438–439.

[9] United States Environmental Protection Agency. 2020. Interactive Map of Air Quality Monitors. **https://www.epa.gov/outdoor-air-quality-data/interactive-map-air-quality-monitors**. (2020).

[10] Junade Ali and Vladimir Dyo. 2017. Coverage and mobile sensor placement for vehicles on predetermined routes: A greedy heuristic approach. (2017).

[11] Yewande Awe, Jostein Nygard, Steinar Larssen, Heejoo Lee, Hari Dulal, and Rahul Kanakia. 2015. Clean air and healthy lungs: enhancing the World Bank's approach to air quality management. (2015).

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

[13] US Census Bureau. 2018. QuickFacts: San Francisco County, California. (accessed August 2019). **https://www.census.gov/quickfacts/fact/table/sanfranciscocountycalifornia,CA,US/PST045218**. (2018).

[14] United States Census Bureau. 2016. 2016 U.S. Gazetteer Files. (accessed August 2019). **https://www2.census.gov/geo/docs/maps-data/data/gazetteer/2016_Gazetteer/2016_gaz_place_06.txt**. (2016).

[15] Tusher Chakraborty, Akshay Uttama Nambi, Ranveer Chandra, Rahul Sharma, Manohar Swaminathan, Zerina Kapetanovic, and Jonathan Appavoo. 2018. Fall-curve: A novel primitive for IoT Fault Detection and Isolation. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. ACM, 95–107.

[16] Badri Chatterjee. 2020. Mumbai to get country's largest air quality monitoring network. **shorturl.at/bjyKM**. (2020).

[17] Dipak K Dash. 2018. India's Public Transport Is In Bad Shape Because It Needs 30 Lakh Buses, But Has Only 3 Lakh. **shorturl.at/pyLS3**. (2018).

[18] 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*. IEEE, 663–667.

[19] 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*. ACM, 29–39.

[20] 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*. IEEE, 1–9.

[21] Yi Gao, Wei Dong, Kai Guo, Xue Liu, Yuan Chen, Xiaojin Liu, Jiajun Bu, and Chun Chen. 2016. Mosaic: A low-cost mobile sensing system for urban air quality monitoring. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

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

[23] 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.

[24] Dorit S Hochbaum. 1997. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. *Approximation Algorithms for NP-Hard Problem* (1997), 94–143.

[25] Central Pollution Control Board (India). 2020. Central Control Room for Air Quality Management - All India. **shorturl.at/enxY6**. (2020).

[26] Eden Strategy Institute and ONG & ONG Pte Ltd. 2018. Top 50 Smart City Governments. (2018).

[27] Jinwei Liu, Haiying Shen, Husnu S Narman, Wingyan Chung, and Zongfang Lin. 2018. A survey of mobile crowdsensing techniques: A critical component for the internet of things. *ACM Transactions on Cyber-Physical Systems* 2, 3 (2018), 18.

[28] Balz Maag, Zimu Zhou, Olga Saukh, and Lothar Thiele. 2017. SCAN: Multi-hop calibration for mobile sensor arrays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 19.

[29] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. 2010. Parknet: drive-by sensing of road-side parking statistics. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 123–136.

[30] 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*. ACM, 323–336.

[31] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14, 1 (1978), 265–294.

[32] Gustavo Niemeyer. 2008. Blog announcing Geohash (accessed August 2019). **https://web.archive.org/web/20080305223755/http://blog.labix.org:80/#post-85**. (2008).

[33] Press Trust of India. 2019. Already 'severe', Delhi's pollution likely to enter 'emergency' zone on Wednesday. **shorturl.at/yGNT3**. (2019).

[34] World Health Organization and others. 2006. *WHO Air quality guidelines for particulate matter, ozone, nitrogen dioxide and sulfur dioxide: global update 2005: summary of risk assessment*. Technical Report. Geneva: World Health Organization.

[35] Kevin P O'Keeffe, Amin Anjomshoaa, Steven H Strogatz, Paolo Santi, and Carlo Ratti. 2019. Quantifying the sensing power of vehicle fleets. *Proceedings of the National Academy of Sciences* 116, 26 (2019), 12752–12757.

[36] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from **https://crawdad.org/epfl/mobility/20090224/cab**. (Feb. 2009). DOI: **http://dx.doi.org/10.15783/C7J010**

[37] C Arden Pope Iii, Richard T Burnett, Michael J Thun, Eugenia E Calle, Daniel Krewski, Kazuhiko Ito, and George D Thurston. 2002. Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution. *Jama* 287, 9 (2002), 1132–1141.

[38] World Population Review. 2019. Rome Population. (accessed August 2019). **http://worldpopulationreview.com/world-cities/rome-population/**. (2019).

[39] Yuecheng Rong, Zhimian Xu, Ruibo Yan, and Xu Ma. 2018. Du-parking: Spatio-temporal big data tells you realtime parking availability. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 646–654.

[40] Olga Saukh, David Hasenfratz, Abouzar Noori, Tamara Ulrich, and Lothar Thiele. 2012. Demo-Abstract: Route Selection of Mobile Sensors for Air Quality Monitoring. In *Proc. of 9th European Conference on Wireless Sensor Networks (EWSN 2012)*. Springer, Trento, Italy.