ConDense: Managing Data in Community-driven Mobile Geosensor Networks

Sebastian Cartier[1], Saket Sathe[1], Dipanjan Chakraborty[2], and Karl Aberer[1]

[1] Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{name.surname}@epfl.ch
[2] IBM Research India
cdipanjan@in.ibm.com

Abstract. Effectively managing the data generated by community-driven mobile geo-sensor networks is a new and challenging problem. One important step for managing and querying sensor network data is to create abstractions of the data in the form of models. These models can then be stored, retrieved, and queried, as required. There has been significant amount of prior literature on using models for query processing [6, 8, 11, 14, 20]. On the contrary, however, there has been a lack of understanding on developing reliable models, considering the unique characteristics of community-driven geo-sensor networks.
In an effort to correct this situation, this paper proposes various approaches for modeling the data from a community-driven mobile geo-sensor network. This data is typically collected over a large geographical area with mobile sensors having uncontrolled or semi-controlled mobility. Therefore, we propose adaptive techniques that take into account such mobility patterns and produce an accurate representation of the sensed spatio-temporal phenomenon. To substantiate our proposals, we perform extensive evaluation of our methods on two real datasets.

1 Introduction

Research in mobile geo-sensor networks is rapidly evolving to investigate the novel paradigm of community-driven sensing. Here, sensors of various sorts (e.g., multi-sensor units monitoring air quality, cell phones, thermal watches, thermometers in vehicles, etc.) are carried by community (public vehicles, private vehicles, or individuals) during their daily activities, collecting data about the environment. In this paper, we present the *ConDense* (Community-driven Sensing of the Environment), a framework for efficiently managing data generated about the environment.

At its core, community sensing is a new form of mobile geo-sensor network [5]. Unique characteristics of this sensing paradigm lie in its organic and unstructured mobile sensing. This is analogous to the Web 2.0 model, where the community participates in generating data. This differs from traditional mobile geo-sensor networks, where the primary objective is to monitor the environment through a controlled specification of desired sampling, mobility characteristics, or through appropriate sensor placement [16, 19].

This work investigates different approaches of *condensing* the data generated by large-scale Community-driven Mobile GeoSensor Networks (CGSN). The *ConDense*

framework takes into account the unique properties of CGNSs and treats the underlying sensor network as a disconnected component, which is collecting data using local policies and principles. Although there is significant literature on model-based query processing on mobile sensor networks, there is a lack of understanding of approaches to determine high quality and concise models of the phenomenon from CGSNs. The models built using the raw data are necessary, since raw data generated from sensors is often, *"imprecise, and erroneous, and hence rarely usable as it is"* [11]. The raw data generated needs to be synthesized and managed for consumption by scientists, applications, and the community.

Regression-based modeling approaches have been proposed in literature to provide mathematically meaningful descriptions of the phenomenon being sensed. For example, [11] presents such a model-based view of sensory readings (temperature in rooms). Here, the applications query only the models, and the models, in turn, get updated as time progresses and new data arrives. However, most prior work implicitly assumes that the sensors are relatively homogeneously distributed and/or their sensing behavior can be tuned, considering the phenomenon being sensed. Typically, trials have used small-scale deployments (e.g., covering a room or a small field).

Unfortunately, CGSNs cannot be tightly controlled and deployments cover large areas (e.g., part of a city or state). Hence, it is difficult to produce a homogeneous, good quality view of the phenomenon. The community-sensing pattern leads to spatio-temporal irregularities in the sensing; while some areas might be adequately sampled, some other areas would not be. A challenging question is: *how do we efficiently create quality-controlled models that cover the sensed data, spatially and temporally?*

Traditional geo-statistical techniques like Kriging [9] can be used for modeling such phenomenon. Kriging interpolates the best linear unbiased estimate of a value at an unobserved point in space, based on the weighted linear combination of surrounding observations, minimizing the approximation error. We found, however, such approaches incur high computational complexity, and hence suffer from scaling issues with dynamic temporal variations. On the other extreme, a naïve strategy would be to grid the area under consideration into equal size grid cells and compute a model per grid cell. This approach is simple, however, might lead to lower quality models.

We propose adaptive strategies that discover spatial areas that can be modeled using single or multiple models. Our strategies adapt to the changing nature of the sensed phenomenon by adjusting the geographical granularity of the models to capture the phenomena with high fidelity. In addition, our strategies have user-defined approximation error thresholds, which can used for adjusting the level of geographical granularity and quality of the models produced by our approaches. On the temporal dimension, we use slack functions (on the models) to time-out low quality models (i.e., models that no longer fit the current data).

To summarize, this paper makes the following contributions:

- We present the *ConDense* data management framework that provides a multi-model based abstraction by condensing information generated by a CGSN.
- To capture the environmental phenomena with high fidelity, we advocate the use of multiple models, which we call *model covers*, for modeling large geographical areas of a CGSN.

– We propose two novel techniques, namely adaptive DBSCAN and adaptive k-means, that adaptively model the data respecting user-defined constraints.
– To establish the efficacy of our approaches, we carry out rigorous experimental evaluation of all the modeling approaches using two real community-sensed datasets.

2 Sensors, Deployment, and Data Collection

In this section we describe the details of the sensor network deployments that are considered in this paper. We consider two sensor network deployments, namely OpenSense and Safecast. In the following section, we discuss the details of the sensors, which are a part of these deployments, and the datasets that are collected for experimental evaluation.

Opensense. The OpenSense [5] project (the main source of funding for this work) currently has two deployments, in the cities of Lausanne and Zurich in Switzerland. In both deployments, the sensors are placed on public transport vehicles, like buses or trams, and additionally include stationary monitoring stations at strategic locations. Fig. 1 shows the infrastructural overview of the OpenSense deployments. The sensors monitor the concentration of various environmental pollutants like, Carbon Monoxide (CO), Carbon Dioxide (CO_2), Nitrogen Dioxide (NO_2), and Ozone (O_3). Table 1 shows the important characteristics of the sensors used for monitoring these pollutants.

The normal urban concentration shown in Table 1 for pollutants is the permissible concentration of a pollutant in an urban environment. These concentrations are given by the National Ambient Air Quality Standards (NAAQS) [1] based in the United States. As will be discussed in Section 4, these normal urban concentration ranges will be used for weighing the approximation error made while approximating the pollutant concentration using a model.

We use the dataset collected from a mobile station mounted on a tram in Zurich, Switzerland. This dataset was collected over seven weeks. For our experimental evaluation, we use the Ozone (O_3) values. The sensors mounted on the tram follow a local sampling policy. An important property of this data is that it was collected from a rela-

Table 1. Characteristics of sensors and pollutants.

Pollutant	Type	Normal Urban Concentration	Average Power
NO_2	electrochemical	0.008 to 0.04 ppm	45 mW
CO	electrochemical	0.5 to 5 ppm	0.85 mW
CO_2	electrochemical	500 to 1500 ppm	0.5 W
O_3	semi-conductor	0.05 to 0.15 ppm	-
Radiation	event counter	0 to 0.23 μSv/h	-

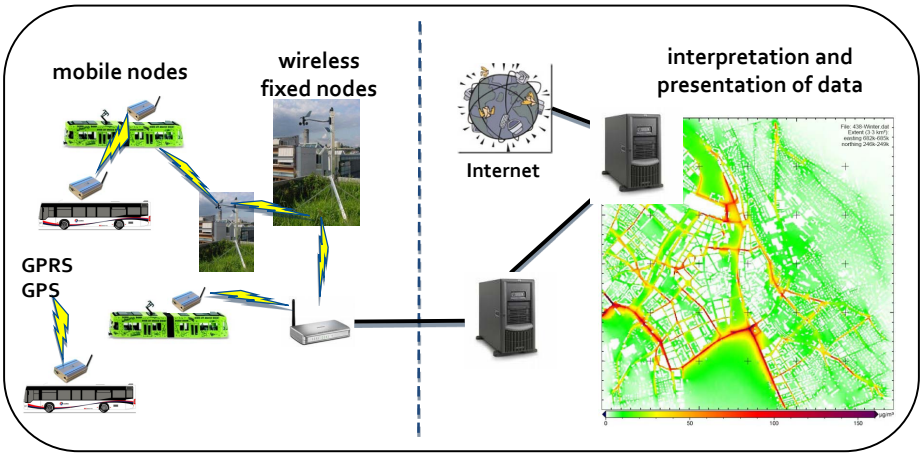


Fig. 1. Community-driven mobile geosensor network infrastructure.

tively clean environment of Zurich, therefore this dataset does not contain large amount of variation in the values of O_3, NO_2, CO_2, etc. We denote this dataset as *opensense*.

Safecast. The Safecast [2] project is a community-driven global sensor network deployment that was kick-started one week after the Fukushima Daiichi nuclear disaster[3] to monitor the radiation level in eastern Japan. The project enables people to both contribute and freely use the collected data. The project is a community-driven project with over one hundred volunteers contributing to the project.

The radiation data is collected by using: (a) 35 mobile stations that are attached to the cars of the volunteers, (b) 50 handheld stations, and (c) 50 static stations. The measurement unit of radiation is micro Sievert per hour (μSv/h). This unit attempts to evaluate the biological effects of radiation as opposed to other radiation units, which just measure the absorbed dose of radiation energy.

Since there are a variety of sensors being used for radiation measurement, the collected data is less accurate as compared to the OpenSense deployment. For our experimental evaluation we use the radiation data from Safecast. This dataset was collected over a period of twenty five weeks. We denote this dataset as *safecast*. Table 2 gives a summary of both the datasets.

3 Related Work

In environmental science, rich models are developed to model environmental phenomenon. For example, air quality [4] models consider three core aspects: pollution sources, transport (wind), and chemical processes. Finally, considering terrain characteristics (e.g. elevation, built-up areas, etc.), models are built to predict expected pollution readings. Appropriate geo-statistical interpolation techniques like Kriging [9] or Gaussian

[3] `http://en.wikipedia.org/wiki/Fukushima_Daiichi_nuclear_disaster`

Table 2. Summary of the Datasets

	opensense	*safecast*
Monitored parameter	Ozone	Radiation Exposure
Number of data values	110,500	970,000
Sensor accuracy	$\pm$2 ppb	-*
Sampling interval	40 sec.	5 sec.

*Radiation counters have variable accuracy.

plumes [18] are used to infer spatio-temporal models of the phenomenon. Validation is carried out using carefully designed sensor layouts, using few high-precision sensors.

While appropriate for visualization or creating rich models from the data, unfortunately, these geo-statistical techniques are unsuitable for modeling the CGSN data in a database environment. This is because they take enormous computation time (e.g., of the order of hours [4]), and hence cannot be applied repeatedly to model error-prone and incomplete data streams from a geographical area. Database environments need to accept incoming sensory data and build models for consumption by queries. To do so, we need solutions that consider performance parameters like model quality, but also account for computational efficiency, query response time, down-time, etc.

In database environments, model-based approaches on distributed sensor data [6,11, 14,22] decouple the sensory updates from the query infrastructure by creating models of the underlying data and allowing the queries to view and operate on top of the models. There are different works that build temporal (per sensor) or spatial models on well-defined regions (for e.g., using grids [11]). Prior work has also suggested in-network modeling [6, 14] to reduce communication overhead.

Such approaches have not considered the ramifications of developing model covers on top of the CGSN data. Firstly, unlike prior deployments, sensors in a CGSN have autonomous (buses) or uncontrolled (private cars) mobility. Hence per-sensor models are inappropriate, since the phenomenon changes behavior as the sensors move over larger areas like cities. Secondly, such approaches have problems with the quality of data. Prior approaches implicitly assume a quasi-uniform distribution of readings for learning the models (e.g., basis function selection or weight optimization). Community sensed data is unevenly distributed (skewed), spatially and temporally. Hence, it is challenging to design methods for quality-controlled model covers that have reasonable performance overhead.

As such, there are many projects today [3,7,10,17,23] exploring community-driven sensing of environmental phenomena. Most of these projects primarily focus on systems issues like developing inexpensive sensors, calibration, how to provide incentives to the users, reduce sampling overhead [15]. None of these projects investigate the research question of exploring efficient strategies to create a model-based data abstraction layer, suitable for database environments.

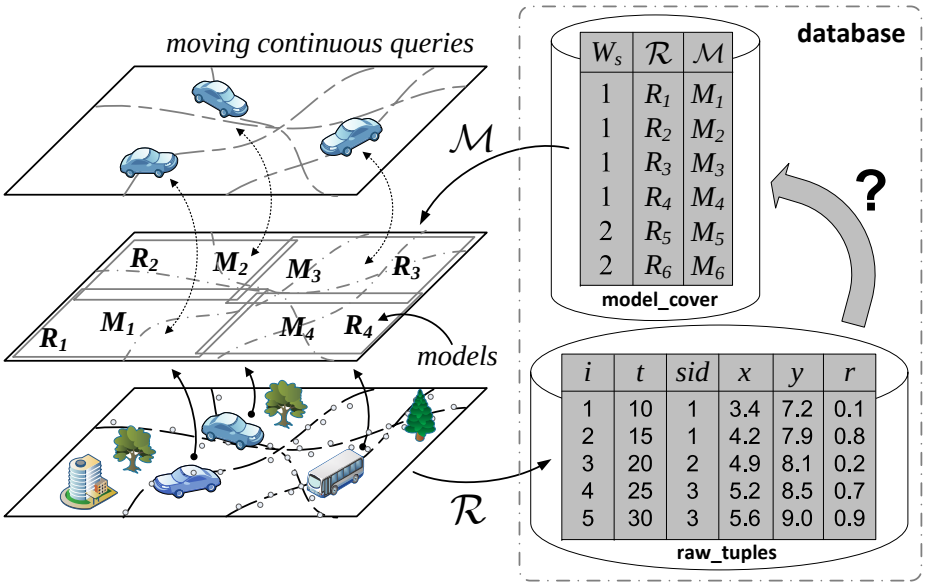


Fig. 2. Architecture of the *ConDense* framework.

4 Problem Characterization

Before diving into the details of the proposed approaches, we present foundational definitions and establish the notation used in the rest of the paper. We start by introducing the overall framework of *ConDense*, which is shown as a schematic in Fig. 2. For simplicity, we decompose this framework into the following three components:

Sensors. This component is responsible for sensing the environment. We assume that there are sensors that are moving over a geographical region $\mathcal{R}$ (refer Fig. 2). For example, $\mathcal{R}$ could be a suburb, city, state, or even a larger geographical area. In addition, we assume that all the sensors have their mobility limited to the region $\mathcal{R}$ and are sensing the parameters of interest. In this paper, we are interested in parameters like pollution and/or radiation.

We assume that the values transmitted by these sensors are continuously updated in a database table called `raw_tuples`. Each tuple i in the table of `raw_tuples` consists of the sensor identifier sid, the time t_i at which the value was sensed, the GPS co-ordinates of the sensed value (x_i, y_i) and the sensor value r_i. Additionally, we denote a single raw tuple in the database as $b_i = (t_i, x_i, y_i, r_i)$, its position as $g_i = (x_i, y_i)$, and its positioned value as $v_i = (x_i, y_i, r_i)$.

Models. The modeling component provides a multi-model abstraction (i.e., model cover) over the raw tuples. On the one hand, it is responsible for answering continuous queries registered by the vehicles; and on the other hand, it is responsible for continuously maintaining the models that are obtained using raw tuples.

Our main objective in this paper is to build and continuously maintain a model cover over the region $\mathcal{R}$. Before proceeding further, let us rigorously define a model cover over the region $\mathcal{R}$.

Definition 1. ***Model Cover.*** *Given a region $\mathcal{R}$ and its partition into regions $R_1, R_2, \ldots, R_p$, such that $\cup_{\alpha=1}^{p} R_\alpha = \mathcal{R}$. A* model cover *is defined as a set of models $\mathcal{M} = \{M_\alpha | 1 \leq \alpha \leq p\}$, where model M_α models the region R_α respectively, for all $1 \leq \alpha \leq p$.*

In this paper, additionally, our objective is also to maintain the model cover as raw tuples are streamed into the system. This task involves adapting the model cover to the changes of the phenomenon that are observed over the region $\mathcal{R}$. To perform these tasks, we should define a temporal dimension of the model cover. Our framework assumes that the model cover is computed using the raw tuples in a time window of length H. Using H, we define a window of raw tuples as $W_s = \langle b_i | sH \leq t_i \leq (s+1)H \rangle$, where s is a positive integer. Thus, W_s is a set of all the raw tuples b_i falling in the interval sH to $(s+1)H$. In addition, we write $g_i \sqsubset W_s$ and $v_i \sqsubset W_s$ to respectively denote the position $g_i = (x_i, y_i)$ and positioned value $v_i = (x_i, y_i, r_i)$ found in the raw tuple $b_i \in W_s$.

In this paper our focus is on estimating a model cover over the region $\mathcal{R}$ for the values in a time window W_s. For clarity, let us concretely define the problem of model cover estimation:

Problem 1. **Model Cover Estimation.** Given the region $\mathcal{R}$ and the window of raw tuples W_s compute the model cover $\mathcal{M}$ such that:

- It partitions/segments $\mathcal{R}$ into p regions $R_1, R_2, \ldots, R_p$ covering the region $\mathcal{R}$,
- It estimates models $M_1, M_2, \ldots, M_p$, such that each corresponds to a region $R_1, R_2, \ldots, R_p$ respectively.

We propose various solutions for solving Problem 1. Broadly, the proposed solutions are of two types: (a) *non-adaptive* solutions that perform the partitioning and estimation steps of Problem 1 only once, without iteratively improving the partitioning; and (b) *adaptive* solutions that perform the partitioning and estimation steps of Problem 1 several times until a user-defined quality criteria (e.g., approximation error) is satisfied. In this paper, we investigate two non-adaptive techniques, then, based on our observations, we propose two time- and space-efficient adaptive techniques that are able to accurately estimate the model cover $\mathcal{M}$ over a large geographical area.

Queries. To make our framework schematic complete, we show the query processing component in Fig. 2. The queries consists of vehicles that register moving continuous queries. An example of such a query registered by a vehicle o_q could be:

Query 1 ***Moving Continuous Query****. Given the position $g = (x, y)$ of the vehicle o_q, continuously return the concentration of NO_2 around o_q at an interval of 10 seconds.*

These queries can be answered directly using the model cover $\mathcal{M}$ [8, 11, 20]. Note that although queries like Query 1 can be directly answered using the table `raw_tuples`, it is neither efficient nor accurate, since: (a) the number of raw tuples could be considerably large as compared to the number of models, and (b) the models minimize the errors

caused during communication or due to the inherent imprecision of the sensors [8, 21]. Remember, query processing is not the primary focus of this paper; nonetheless, this component is shown in Fig. 2 for presenting a complete picture of the *ConDense* framework.

Error Metric. The last foundational aspect is the error metric that we use in this paper. Consider a model cover estimation method that partitions the window W_s into regions R_α where $1 \leq \alpha \leq p$, such that W_s^α denotes the set of raw tuples b_i that are in region R_α. Suppose the model M_α approximates the value r_i with $\bar{r}_i$ then the error metric is defined as:

$$u_\alpha = \frac{100}{|W_s^\alpha|} \sum_{v_i \sqsubset W_s^\alpha} u_\alpha(v_i), \quad u_\alpha(v_i) = \frac{|r_i - \bar{r}_i|}{n_{high} - n_{low}}, \tag{1}$$

where n_{high} and n_{low} are the upper and lower bounds of the normal concentration of the measured pollutant found in the urban environment. For example, if we are measuring Ozone, then the normal concentration of Ozone in normal urban environment is $n_{high} = 0.15$ ppm and $n_{low} = 0.05$ ppm (refer Table 1). Thus, the error metric shown in (1) measures the relative error as compared to the normal range of values. We call this error metric the *normal percentage error*[4]. Intuitively, the normal percentage error is a fair error metric as compared to the other error metrics because it quantifies the impact of the error on the environment.

5 Non-Adaptive Methods for Model Cover Estimation

In this section we present the non-adaptive model cover estimation methods. Specifically, we investigate to strategies: first, a naive strategy in which the partitioning of $\mathcal{R}$ is performed by a rectangular division, second, we discuss a largely popular technique from the geo-statistics literature called Kriging. We observe that the non-adaptive methods are either computational expensive or inaccurate. In addition, as will be seen later, storing the model cover generated by these methods is also considerably expensive.

5.1 Grid-based Model Cover

The Grid-based (GRIB) model cover estimation method is the most naïve strategy for estimating a model cover. This approach involves overlaying a grid over the region $\mathcal{R}$ and then estimating a linear regression model for individual grid elements. It simply divides the region $\mathcal{R}$ into a grid of a fixed size $n \times n$. Then each grid element forms the region R_α from Definition 1, such that $p = n^2$. Now the set of regions $R_1, R_2, \ldots, R_p$ induce a partition on the raw tuples in the window W_s. Let us denote the set of raw tuples of the window W_s contained in the region R_α by W_s^α. Now we can estimate a linear regression model M_α over the values W_s^α as:

$$r_i = \bar{r}_i + e_i \quad \bar{r}_i = a_0 + a_1 x_i + a_2 y_i. \tag{2}$$

[4] We use *normal percentage error* and *approximation error* interchangeably.

Here, we estimate the parameters (a_0, a_1, a_2) by performing a least-squares fitting that minimizes the sum of e_i^2. The interpolation of the value at a position $g' = (x', y')$ is performed as:

$$\hat{r}(g') = a_0 + a_1 x' + a_2 y'. \tag{3}$$

The main advantage of the GRIB model cover estimation method is that it is simple to implement. This simplicity comes from the static nature of the partitioning scheme; the partitioning scheme does not consider the characteristics of the underlying data. In the GRIB method, the granularity of the partitioning does not evolve temporally. Especially, for large geographical areas there could be a need to dynamically change the granularity and size of the partitioning based on the nature of the underlying phenomenon. For example, during peak hours of traffic, pollution is higher in downtown areas as compared to residential areas, and therefore we need a partitioning scheme that adapts to such change in behavior.

5.2 Kriging-based Model Cover

The Kriging-based (KRIB) model cover estimation method is an approach that involves the use of Kriging [9]. Kriging is a well-known geo-statistical method for producing highly accurate models of data. In comparison to other interpolation approaches, Kriging has the advantage that it can also assign a confidence value to the interpolated values. These advantages (high accuracy and confidence values) naturally invite additional cost for creating and querying a Kriging-based model cover.

Kriging interpolates the value at position $g' = (x', y')$ by summing the weighted known values r_i as follows:

$$\hat{r}(g') = \sum_{i=1}^{|W_s|} \lambda_i r_i, \quad \gamma(g_i, g') = \sum_{j=1}^{|W_s|} \lambda_i \gamma(g_i, g_j), \tag{4}$$

where λ_i are the weights, such that $\sum_{i=1}^{|W_s|} \lambda_i = 1$ and $\gamma(g_i, g_j)$ is the semi-variogram of the points g_i and g_j. λ_i are evaluated by solving the set of equations for $\gamma(g_i, g')$ where $1 \leq (i, j) \leq |W_s|$. Additional details regarding Kriging can be found in [9].

Query processing time can be reduced by pre-computing the inverse matrix formed by $\gamma(g_i, g_j)$. Since $\gamma(g_i, g_j)$ has size $|W_s| \times |W_s|$, storing the inverse of $\gamma(g_i, g_j)$ requires a large amount of memory. In Section 7, we show that even with pre-computation of the inverse of $\gamma(g_i, g_j)$, the KRIB model cover estimation method is not comparable with other model cover estimation approaches in answering point (interpolation) queries.

The Kriging method was introduced to efficiently approximate values when the sensors are stationary. But this method is not well suited for moving sensors, since in a mobile sensing environment the values along hotspots are excessively dense and should be condensed to reduce redundant sampling. Secondly, Kriging tries to fit a function to all the sensed values without eliminating redundant information, and, therefore has large overhead in terms of storage and computational complexity.

6 Adaptive Methods for Model Cover Estimation

In contrast to the non-adaptive techniques discussed in Section 5, the methods proposed in this section exploit the characteristics of the underlying data for obtaining a better partitioning of $\mathcal{R}$. In Section 7, we thoroughly compare the adaptive and non-adaptive methods, and experimentally establish the superiority of the adaptive techniques. Our adaptive techniques are based on unsupervised clustering algorithms. They intelligently partition $\mathcal{R}$ into regions, such that the models are always able to approximate the data with a certain error guarantee.

6.1 Adaptive DBSCAN

The *adaptive DBSCAN* method is based on the well-known DBSCAN algorithm proposed in [12]. Let us begin by first understanding the DBSCAN algorithm, then we briefly discuss the reasons for the unsuitability of the DBSCAN algorithm for our problem; followed by the description of the adaptive DBSCAN method.

DBSCAN. Given a window of raw tuples W_s, DBSCAN defines the density of $g_i \sqsubset W_s$, denoted as $N_{Eps}(g_i)$, as the number of points that are present in a radius Eps around g_i. $g_i \sqsubset W_s$ is called a *core point* if $N_{Eps}(g_i)$ is greater than $MinPts$, where $MinPts$ is a user-defined constant. All the points around g_i present in a radius Eps are called *directly density-reachable* from g_i.

A position g_j is *density-reachable* from g_i if there is a chain $(g_*)_1, \ldots, (g_*)_l$, where $(g_*)_1 = g_i$ and $(g_*)_l = g_j$, such that $(g_*)_2$ is directly density-reachable from $(g_*)_1$, $(g_*)_3$ from $(g_*)_2$, so on until $(g_*)_l$. Two positions g_i and g_j are *density-connected* if they are both density-reachable from a core point g_c. Now, we define W_s^α as a set of raw tuples, where $g_i \sqsubset W_s^\alpha$ is density-connected with $g_j \sqsubset W_s^\alpha$ for all $i \neq j$.

If a position g_i is not density-connected with any other points in W_s, it is considered as noise and we set $c_i = NOISE$, where c_i represents the cluster membership of a raw tuple b_i. By randomly selecting unclustered points (i.e., points where $c_i = UNCLASSIFIED$) and clustering all density-reachable tuples into the same region W_s^α we can divide the set W_s into k_a regions, where $0 \leq k_a \leq (|W_s|/MinPts)$.

DBSCAN clusters the raw tuples only based on g_i and does not consider the sensor values r_i. Thus, it is possible that DBSCAN produces regions that cannot be modeled using polynomials having lower number of coefficients. To rectify this situation, we modify the DBSCAN algorithm such that it produces regions that can be modeled using lower number of coefficients. We call this modified algorithm *Adaptive DBSCAN*.

Adaptive DBSCAN. In the Adaptive DBSCAN (Ad-DBS) method we continuously maintain a linear regression model M_α (refer (2)) for each region R_α. In addition, we provide the following modified definition for density-reachable and density-connected:

Definition 2. ***Model Density-reachable.*** *A positioned value v_i is model density-reachable from $v_j \sqsubset W_s^\alpha$, if position g_i is density-reachable from g_j and $u_\alpha(v_j) < \tau_r$, where τ_r is a user-defined quality threshold.*

Definition 3. ***Model Density-connected.*** *Positioned value v_i and v_j are model density-connected if v_i and v_j are model density-reachable from $v_\ell \sqsubset W_s^\alpha$.*

Algorithm 1 The adaptive DBSCAN algorithm.

Input: Window W_s, error threshold τ_r, Eps, $MinPts$.

Output: Number of regions k_a, regions R_α and a linear regression model M_α for each region respectively where $\alpha = 1, \ldots, p$.

1: $k_a \leftarrow 1$
2: **for all** $v_i \sqsubset W_s$ **do**
3: **if** $c_i = UNCLASSIFIED$ **then**
4: **if** EXPANDCLUSTER(v_i,k_a) **then**
5: $k_a \leftarrow k_a + 1$
6: **procedure** EXPANDCLUSTER(v_i,k_a) : boolean
7: $seeds \leftarrow$ REGIONSEARCH$(v_i, Eps) \setminus v_i$
8: **if** $|seeds| < MinPts$ **then**
9: $c_i \leftarrow NOISE$
10: **return** $false$
11: **else**
12: ADD(M_{k_a},v_i)
13: **for all** $v_j \in seeds$ **do**
14: **if** CHECKERRORANDADD(M_{k_a},v_j) $\neq$ success **then**
15: $seeds \leftarrow seeds \setminus v_j$
16: **while** $|seeds| \neq 0$ **do**
17: $v_j \leftarrow removeOneValue(seeds)$
18: $results \leftarrow$ REGIONSEARCH$(v_j, Eps) \setminus v_j$
19: **if** $|results| > MinPts$ **then**
20: **for all** $v_h \in results$ **do**
21: **if** $c_h = UNCLASSIFIED$ **then**
22: $seeds \leftarrow seeds \cup c_h$
23: **if** $c_h \in \{NOISE, UNCLASSIFIED\}$ **then** CHECKERRORANDADD(M_{k_a},v_h)
24: **return** true

Algorithm 1 performs the partitioning of W_s, such that each positioned value $v_i \sqsubset W_s^\alpha$ is model density-connected to $v_j \sqsubset W_s^\alpha$ for all $i \neq j$. The function CHECKERRORANDADD temporarily adds v_j to W_s^α and re-computes the model M_α. If $u_\alpha(v_j) > \tau_r$, then v_j is not model density-connected to the other tuples in W_s^α, therefore it is not permanently added to W_s^α. In Step 7, REGIONSEARCH return the points in a radius Eps around v_i, and in Step 12, ADD unconditionally adds v_i to M_α.

Interpolation using Ad-DBS. Because of the new definitions of model density-reachable and density-connected it may happen that the regions R_α produced by the Ad-DBS method overlap with each other. Therefore, for interpolating the value $\hat{r}(g')$ at position (x', y') it is unclear whether one or more regions R_α should be used. To solve this problem, we introduce a weighing scheme (refer Fig. 3) that produce the interpolated value $\hat{r}(g')$ by assigning weighting functions $K_\alpha(g')$ to the regions R_α, such that:

$$\hat{r}(g') = \sum_{\alpha=1\ldots p} \kappa_\alpha(g')\hat{r}_\alpha(g'), \tag{5}$$

where $\kappa_\alpha(g') = \frac{K_\alpha(g')}{\sum_{\beta=1\ldots p} K_\beta(g')}$ and $\hat{r}_\alpha(g')$ is the interpolated value using model M_α.

Since the normal percentage error metric introduced in (1) does not consider overlapping regions, we introduce the following modified version of the normal percentage error for analyzing this weighting scheme of the Ad-DBS method:

$$\hat{u}_\alpha = \frac{100}{|W_s^\alpha|} \sum_{v_i \sqsubset W_s^\alpha} \hat{u}_\alpha(v_i), \quad \hat{u}_\alpha(v_i) = \frac{|r_i - \hat{r}_i(g_i)|}{n_{high} - n_{low}}. \tag{6}$$

Notably, the difference between u_α and $\hat{u}_\alpha$ characterizes the error introduced by the weighting scheme used in the Ad-DBS method.

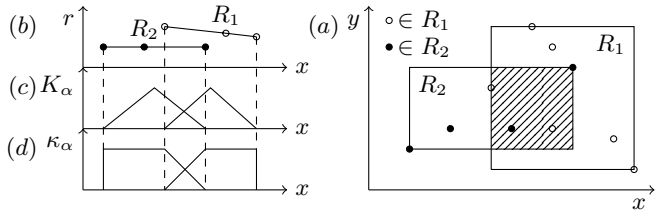


Fig. 3. (a) shaded area shows an example of two overlapping regions, (b) shows the regions with the corresponding sensor values r, (c) and (d) present the weighting functions K_α and κ_α used for interpolation.

6.2 Adaptive K-Means

In this section we start by discussing the k-means clustering method, then briefly discuss the reasons why the vanilla k-means clustering method cannot be used for obtaining a model cover with a user-defined approximation error threshold. Then we propose the adaptive k-means model cover estimation method, that overcomes the shortcomings of the the k-means clustering method and efficiently produces an highly accurate model cover.

K-means Clustering. Given the raw tuples in a window W_s and the number of clusters k, the objective of the k-means clustering method is to divide the raw tuples in the window W_s into k sets $W_s^1, W_s^2, \ldots, W_s^k$ such that the following objective function is minimized:

$$\arg\min_{\hat{\mu}_\alpha} \sum_{\alpha=1}^{k} \sum_{v_j \sqsubset W_s^\alpha} ||v_j - \hat{\mu}_\alpha||, \tag{7}$$

where $\hat{\mu}_\alpha = (x_\alpha, y_\alpha, r_\alpha)$ is known as the centroid of the region partition W_s^α. Then the region R_α is the region that surrounds points W_s^α, and the model cover can be obtained by computing a linear regression model M_α for each W_s^α.

The k-means clustering method does not achieve our objective of partitioning the raw values W_s, since the euclidean distance used by the k-means method may compensate a large difference in the sensor value r with a small difference in the position

(x, y). On the contrary, our objective is that values in a particular region R_α should be close in the position *and* in the sensor value. Moreover, another requirement is that the raw tuples W_s should be approximated within a user-defined normal percentage error threshold τ_n. For achieving these objectives we propose an adaptive variant of the k-means clustering method.

Adaptive K-means. The algorithm used by adaptive k-means (Ad-KMN) method is shown in Algorithm 2. Fig. 4 shows an example of the Ad-KMN method on toy data.

Assume that before executing the Ad-KMN method, we compute two k-means centers μ_1 and μ_2 over all the positions $g_i \sqsubset W_s$. A snapshot after this step is shown in Fig. 4(*a*). Next, we check whether the errors u_1 and u_2 are within a user-defined threshold τ_n. The principle here is to introduce an additional cluster centroid ω_i for each region R_i where $u_i > \tau_n$, by choosing the g_i that produced the worst error for R_i.

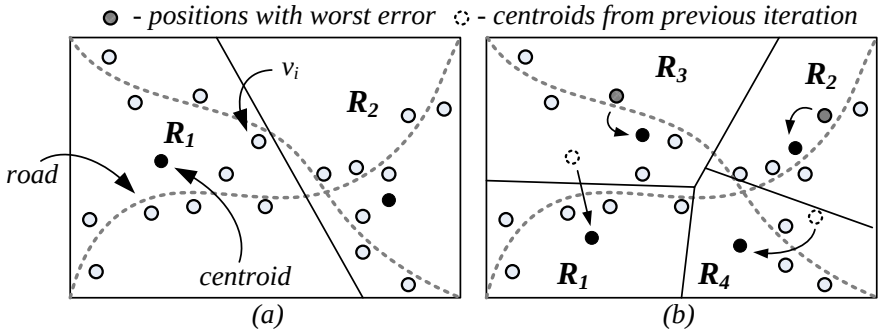


Fig. 4. Ad-KMN iterations on toy data. (a) the centroids of regions R_1 and R_2 are computed, after which models M_1 and M_2 are estimated. (b) since error $u_1 > \tau_n$ and $u_2 > \tau_n$, we add two new clusters R_3 and R_4 using k-means clustering algorithm.

Suppose, both R_1 and R_2 of Fig. 4(*a*) violate the error condition (i.e., $u_1 > \tau_n$ and $u_2 > \tau_n$), then we initialize two new centroids ω_1 and ω_2 and we re-adjust the four centroids (μ_1, μ_2, $\mu_3 = \omega_1$ and $\mu_4 = \omega_2$), by executing the standard k-means algorithm on the four centroids. The result of this step is shown in Fig. 4(*b*).

As will be shown in Section 7, the Ad-KMN method exhibits fast convergence characteristics. In addition, the Ad-KMN method also requires lower storage space and can produce accurate model covers.

6.3 Efficiently Maintaining the Model Cover

Furthermore, we are interested in maintaining the model cover as new windows W_s are streamed into *ConDense*. Specifically, given several windows of raw values W_s where $s = (1, 2, \ldots, S)$, we are interested in continuously maintaining the model cover while minimizing the number of additional computations required for model cover maintenance.

We start by estimating the cluster centroids μ_α over a training window W_D of size $D \gg H$ using the Ad-KMN method. The Ad-KMN method returns the regions R_α and

models M_α where $\alpha = (1, \ldots, k_a)$. Now, assume that the first window of raw values W_1 is available. W_1 is first partitioned according to the cluster centers μ_α, such that W_1^α contains the raw tuples where $||g_i - \mu_\alpha||$ is minimal.

Next, we compute the metric u_α for W_1^α. If u_α is greater than a user-defined threshold τ_r, then we invalidate the model M_α and re-estimate its coefficients. We perform a similar test for all the other W_1^α. We use flops[5] to measure the cost of updating the model M_α. Suppose the cost of updating the window W_s be denoted as $\mathcal{C}(W_s)$, then it can be computed as follows [13]:

$$\mathcal{C}(W_s) = \sum_{\forall \alpha \ s.t. \ u_\alpha > \tau_r} 2 \cdot |M_\alpha|^2 \left(|W_s^\alpha| - \frac{|M_\alpha|}{3} \right), \tag{8}$$

where $|M_\alpha|$ is the number of coefficients to estimate for the model M_α. In our case, $|M_\alpha| = 3$ since M_α has three coefficients (a_0, a_1, a_2). The better the Ad-KMN method partitions the region $\mathcal{R}$, the less would be the cost of maintaining the model cover, since the Ad-KMN method would have found similar areas. Therefore, for raw tuples that are newly streamed into the system in a reasonably short interval do not require a model update, resulting in potentially dramatic saving of computation required for model cover maintenance.

Algorithm 2 The adaptive k-means model cover method.

Input: Window W_s, error threshold τ_n.

Output: Number of regions k_a, regions R_α and a linear regression model M_α for each region respectively where $\alpha = 1, \ldots, k_a$.

```
1: newCluster ← true
2: clusterChanged ← true
3: while newCluster do
4:     newCluster ← false
5:     while clusterChanged do
6:         clusterChanged ← false
7:         for α in 1 to k_a do
8:             μ_{α,n} ← recenter(W_s^α)
9:             if W_s^α ≠ W_s^{α,n} then
10:                clusterChanged ← true
11:            μ_α ← μ_{α,n}
12:    for α in 1 to k_a do
13:        M_α, u_α, ω_α ← estimateModel(W_s^α)
14:        if u_α > τ_n then
15:            k_a ← k_a + 1, μ_{k_a} ← ω_α
16:            newCluster ← true
```

As we will show in Section 7, such a strategy of maintaining the Ad-KMN model cover is effective and can yield up to *approximately 3x less number of flops* as com-

[5] A *flop* represents either the addition or the multiplication of two floating point numbers.

pared to using the same strategy over a GRIB model cover. Thereby, establishing the advantages of using an adaptive method for model cover estimation.

7 Experimental Evaluation

In this section we perform extensive experimental evaluation of the various model cover estimation approaches. In Section 7.1 we compare the model cover estimation approaches with respect to the normal percentage error. In Section 7.2, we compare the efficiency of the adaptive and non-adaptive techniques for model cover estimation in terms of the storage space and estimation time. Lastly, Section 7.3 compares adaptive and non-adaptive methods with respect to their temporal model cover validity characteristics. For all the experiments we use the *opensense* and the *safecast* datasets described in Section 2.

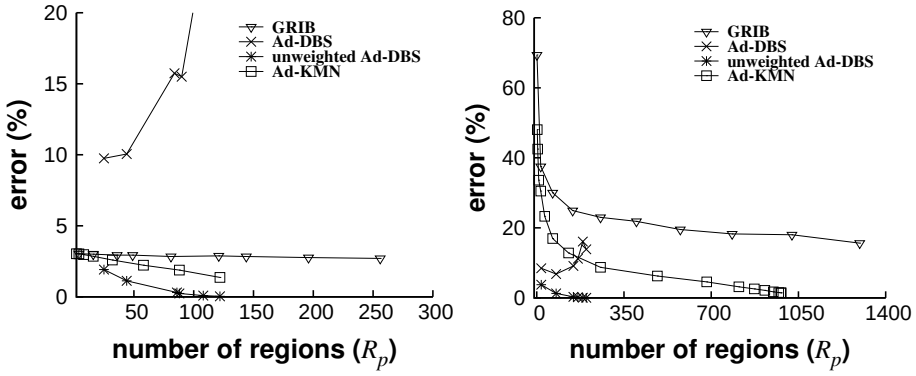
7.1 Error Analysis

We start by analyzing the different model cover estimation approaches using the normal percentage error (1). Fig. 5 shows the error as the number of regions are increased for the GRIB, Ad-KMN, and Ad-DBS methods. The process of adding more regions terminates when the error is less than the user-defined error threshold $\tau_n = 1\%$ or adding new regions does not significantly reduce the error. For this experiment the size of the window W_s is set to 6 hours. Clearly, for all the three approaches the percentage normal error decreases with increase in the number of regions.

Specifically, for *safecast* the Ad-KMN method delivers an *improvement of 12.5 times less error* as compared to the GRIB method for $p = 1000$. In contrast, for *opensense*, the Ad-KMN method does not show significant improvements (2.1 times less error for $p = 120$) over the GRIB method. This is because, as described earlier, *opensense* data does not exhibit high spatial-temporal variation. Therefore all the methods are able to achieve lower error. In general, the adaptive methods have lower number of regions as compared to the non-adaptive methods. Consider *safecast*, at convergence the GRIB method has 1296 regions as compared to the 981 regions of the Ad-KMN method (refer Fig. 5).

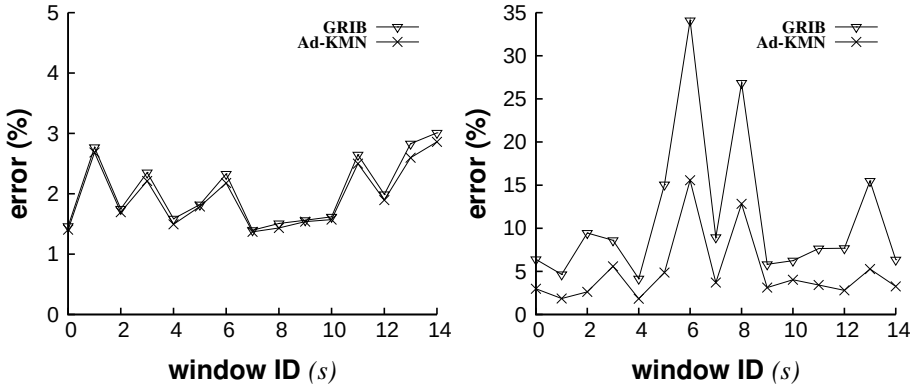
Additionally, to substantiate the results in Fig. 5, we plot the error for 15 randomly chosen windows W_s for the Ad-KMN and GRIB methods where the maximum number of regions $p = 50$ and is constant. Similar observations to Fig. 5 could be made in Fig. 6, for *safecast* the improvement obtained by using the Ad-KMN method as compared to the GRIB method is significantly higher than *opensense*. In Fig. 6, we do not show the result for the Ad-DBS method. Since for the Ad-DBS method, it is impossible to control the number of regions that will be created, thus leading to an unfair comparison. These experiments clearly establish that adaptive methods, like the Ad-KMN method, can dramatically reduce the error as compared to the non-adaptive methods.

Note that Fig. 5 does not show the KRIB method, since the KRIB method always produces zero error due to the fact that Kriging always finds a function that passes perfectly thorough the given points. The zero error of Kriging comes at a cost: estimating



(a) *opensense* (b) *safecast*

Fig. 5. Comparing the decrease in percentage error as the number of regions increase. *Unweighted Ad-DBS* denotes Ad-DBS without the weighting scheme of (5). Note the different ranges on the y-axis.



(a) *opensense* (b) *safecast*

Fig. 6. Comparing the percentage normal error for Ad-KMN and GRIB over randomly chosen windows W_s. Note the different ranges on the y-axis.

and storing a Kriging model is substantially inefficient (refer Section 7.2) as compared to the adaptive methods, and therefore is not suitable for a database environment.

In Fig. 5 the Ad-DBS method produces higher error as compared to the Ad-KMN method. The reason for such behavior is that, the increase in error due to the over-simplified weighing scheme of the Ad-DBS method (see (5)), is more as compared to the decrease in error obtained by adding more regions; thus leading to an overall error increase. To experimentally establish this observation, in Fig. 5 we also show the normal percentage error obtained by the Ad-DBS method without the weighting scheme of (5). This shows that the Ad-DBS method could compete with the other methods if a more sophisticated weighting scheme is introduced.

7.2 Comparing Efficiency of Model Cover Estimation Methods

Next, we compare the time- and space-efficiency of the model cover estimation methods. Fig. 7(a) compares the average time required for model cover estimation using different methods. Fig. 7(b) compares the average time required for processing a point query. Here a *point query* is defined as a query that requests for the interpolated value at a particular position $g = (x, y)$. The average point query processing time is computed over 4000 point queries.

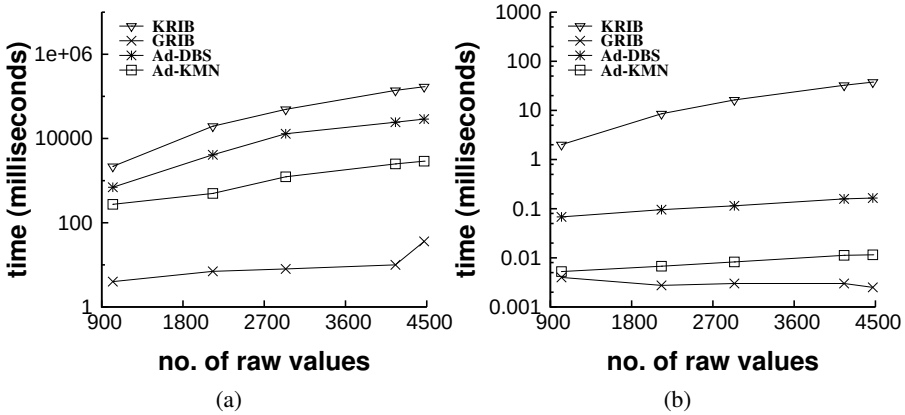


Fig. 7. Comparing efficiency of (a) model cover estimation and (b) processing a point query (interpolation) on *opensense*.

On the one hand, the most time-efficient method for model cover estimation is the GRIB method, on the other hand it is significantly inefficient in terms of space (refer Fig. 8). Moreover, the Ad-KMN method requires 1160 times less memory as compared to the GRIB method, and can be estimated by spending on an average 1.5 seconds more than the GRIB method.

Obviously, the KRIB method is significantly time- and space-inefficient as compared to the other model cover estimation methods, demonstrating that the KRIB method is clearly not usable in a database environment. Lastly, the Ad-DBS method can be stored using slightly less memory, but exhibits minor inefficiency in processing a point query as compared to the Ad-KMN method, and as seen in Section 7.1 it produces high normal percentage error.

7.3 Analyzing Temporal Validity of Model Cover

We perform the last set of experiments on *opensense*. These experiments are performed to compare the temporal validity characteristics between adaptive and non-adaptive model cover estimation methods. Particularly, we are interested to compare temporal behavior of the GRIB and the Ad-KMN methods.

We start by choosing a region $\mathcal{R}' \subset \mathcal{R}$. From the raw tuples in $\mathcal{R}'$, we choose a training window W_D of size 6 hours and 88 testing windows W_s of size 30 minutes. Note that W_D and W_s are consecutive in time. Then we choose the model retain threshold (τ_r) as 1% and apply the algorithm for maintaining the temporal validity of the models from Section 6.3 and compute the cost $\mathcal{C}(W_s)$ for each window W_s. To substantiate our experiment, we choose three different values of p for the Ad-KMN method and adjust the GRIB method so that the number of used grid cells by the GRIB method are always equal to that of the Ad-KMN method.

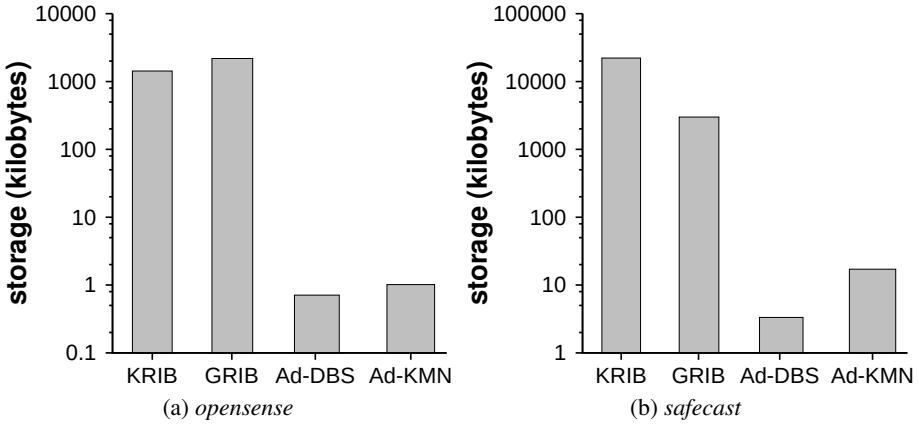
Fig. 9 shows the cumulative number of flops required to maintain the model cover. Admittedly, the Ad-KMN method requires a factor 2.7 less number of flops as compared to the GRIB method. In conclusion, the regions R_α that are produced by the Ad-KMN method are valid for a longer time, thus require less number of flops. For example, the Ad-KMN method requires zero flops for the first 34 windows as opposed to the 1874 flops required by the GRIB method.

8 Conclusions

This paper presents non-adaptive and adaptive techniques for managing data produced by a CGSN. Our experiments establish that the adaptive model cover estimation methods, which use dynamic partitioning approaches, demonstrate promising performance gains as compared to the non-adaptive methods. Particularly promising is the adaptive k-means model cover estimation method, since it shows the best model cover quality, considering other parameters, like storage, computational cost, and temporal validity.

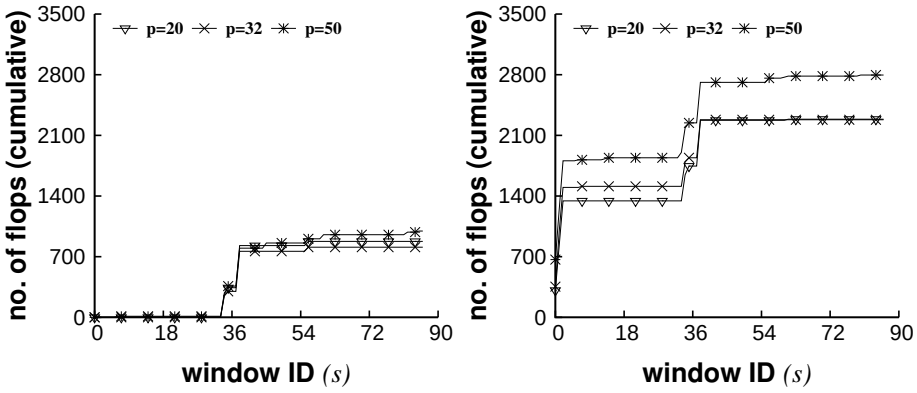
There are many issues that remain to be researched. Following are a few directions we intend to pursue in our future works:

– *Complete re-learn of model cover.* In our approach for handling temporal evolution of the model cover (refer Section 6.3) we have not considered a complete re-learn



(a) *opensense* (b) *safecast*

Fig. 8. Comparing the memory requirement of all the model cover estimation methods.



(a) Ad-KMN (b) GRIB

Fig. 9. Comparing temporal validity of the model cover produced by (a) Ad-KMN and (b) GRIB on *opensense*.

of the model cover if the cost $\mathcal{C}(W_s)$ increases dramatically. On the one hand, re-learning could reduce the cost $\mathcal{C}(W_s)$ for future windows W_s, but on the other hand, could incur down-time for the system. Another alternative to complete re-learn is to develop techniques that merge/split the models, such that a reasonable model cover is always maintained. We plan to explore the trade-off between complete re-learn and merge/split in our future works.

– *Continuous query processing.* The *ConDense* framework describes the continuous query processing component, and evaluates query costs with respect to model cover techniques. As a next natural step, we plan to investigate efficient and accurate query processing solutions. This, we believe, will open-up interesting research issues like, query optimization, response caching, model cover indexing, etc.
– *Utility-driven sampling.* Finally, if we relax the autonomous sensing assumption in the community sensing paradigm then there is a issue of utility-driven sampling. Here, the underlying phenomenon is sampled only as much as required by a given set of continuous queries. The utility is defined by the queries based on the accuracy guarantee requirements provided by the user.

References

1. National Ambient Air Quality Standards (NAAQS). `http://www.epa.gov/air/criteria.html`
2. The Safecast project. `http://blog.safecast.org/`
3. Urban Atmosphere Project. `http://www.urban-atmospheres.net/` (2006)
4. AERMOD (EPA). http://www.epa.gov/scram001/dispersion_prefrec.htm (2009)
5. Aberer, K., Sathe, S., Chakraborty, D., Martinoli, A., Barrenetxea, G., Faltings, B., Theile, L.: OpenSense: Open community driven sensing of environment. In: IWGS (along with ACM GIS) (2010)

6. Bhattacharya, A., Meka, Singh, A.: MIST: Distributed indexing and querying in sensor networks using statistical models. In: VLDB (2007)
7. Brewer *et al.*, E.: N-Smarts: Networked suite of mobile atmospheric real-time sensors. http://www.cs.berkeley.edu/~honicky/nsmarts/ (2007)
8. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: SIGMOD. pp. 551–562 (2003)
9. Chiles, J., Delfiner, P.: Geostatistics: modeling spatial uncertainty. Wiley-Interscience (1999)
10. Costa *et al.*, B.: Air Project. http://www.pm-air.net/ (2006)
11. Deshpande, A., Madden, S.: MauveDB: Supporting model-based user views in database systems. In: SIGMOD (2006)
12. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining. vol. 1996, pp. 226–231. AAAI Press (1996)
13. Golub, G., Van Loan, C.: Matrix computations. The Johns Hopkins University Press (1996)
14. Guestrin, C., Bodik, P., Thibaux, R., Paskin, R., Madden, S.: Distributed regression: An efficient framework for modeling sensor network data. In: IPSN (2004)
15. Krause *et al.*, A.: Towards community sensing. In: IPSN (2008)
16. Liu, C., Wu, K., J., P.: An energy-efficient data collection framework for wireless sensor networks by exploiting spatio-temporal correlation. TPDS 18(7) (2007)
17. Luo *et al.*, L.: Sharing and exploring sensor streams over geocentric interfaces. In: GIS (2008)
18. Miller, C., Hively, L.: A review of validation studies for the Gaussian plume atmospheric dispersion model. In: Journal of Nuclear Safety Vol. vol. 28 (2009)
19. Nittel, S.: A survey of geosensor networks: advances in dynamic environmental monitoring. In: Sensors (2009)
20. Ré, C., Letchner, J., Balazinksa, M., Suciu, D.: Event queries on correlated probabilistic streams. In: SIGMOD. pp. 715–728 (2008)
21. Sathe, S., Jeung, H., Aberer, K.: Creating probabilistic databases from imprecise time-series data. In: ICDE. pp. 327–338 (2011)
22. Thiagarajan, A., Madden, S.: Querying continuous functions in a database system. In: SIGMOD. pp. 791–804 (2008)
23. Willett *et al.*, W.: Common sense community: scaffolding mobile sensing and analysis for novice users. In: Pervasive Computing (2010)