BuildSense: Accurate, cost-aware, fault-tolerant monitoring with minimal sensor infrastructure

RACHEL CARDELL-OLIVER, The University of Western Australia, Australia
CHAYAN SARKAR, TCS Research & Innovation, India

Buildings can achieve energy-efficiency by using solar passive design, energy-efficient structures and materials, or by optimizing their operational energy use. In each of these areas, efficiency can be improved if the physical properties of the building along with its dynamic behavior can be captured using low-cost embedded sensor devices. This opens up a new challenge of installing and maintaining the sensor devices for different types of buildings. In this article, we propose BuildSense, a sensing framework for fine-grained, long-term monitoring of buildings using a mix of physical and virtual sensors. It not only reduces the deployment and management cost of sensors but can also guarantee accurate and fault-tolerant data coverage for long-term use. We evaluate BuildSense using sensor measurements from two rammed-earth houses that were custom-designed for a challenging hot-arid climate so that almost no artificial heating or cooling is required. We demonstrate that BuildSense can significantly reduce the cost of permanent physical sensors whilst still achieving fit-for-purpose accuracy, fault-tolerance, and stability. Overall, we were able to reduce the cost of a building sensor network by 60% to 80% by replacing physical sensors with virtual ones while still maintaining accuracy of $\leq 1.0^{\circ}$C and fault-tolerance of 2 or more predictors per virtual sensor.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*.

Additional Key Words and Phrases: energy-efficient building, sensor data estimation, virtual sensing, sensing as a service

ACM Reference Format:
Rachel Cardell-Oliver and Chayan Sarkar. 2018. BuildSense: Accurate, cost-aware, fault-tolerant monitoring with minimal sensor infrastructure. *ACM Trans. Sensor Netw.* 0, 0, Article 0 (2018), 23 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Residential and commercial buildings account for almost 21% and 18% of total U.S. energy consumption, respectively [26]. This has a direct impact on greenhouse gas emission, thus on environmental health. Emission from the buildings can be reduced either by reducing emissions from the energy supply (e.g., energy generation through renewable sources) or by reducing energy consumption through improved building design and lower energy use [5, 26].

A large body of work aims to make buildings more energy-efficient through better utilization of the HVAC [2, 13] and lighting systems [10, 22]. These systems depend on continuous data collection by a large number of sensors and an established physical model of the buildings. On the other hand, new building designs and new building materials are also investigated to reduce energy consumption. However, there is a lack of scientific evidence about the long-term performance of

Authors' addresses: Rachel Cardell-Oliver, The University of Western Australia, Perth, Australia, rachel.cardell-oliver@uwa.edu.au; Chayan Sarkar, TCS Research & Innovation, Kolkata, India, sarkar.chayan@tcs.com.

1550-4859/2018/0-ART0 $15.00
https://doi.org/0000001.0000001

new building designs in real-world settings [9]. For example, a long term study of the houses built from rammed-earth showed that the state-of-the-art building models significantly underestimated performance and subsequent comfort, and thus over-estimated energy use [5]. Understanding the performance of these new building designs requires long term, fine-grained sensor data measured in situ.

Problem description. With the advent of embedded technologies, automated data collection using embedded sensor devices has become mainstream. To understand an indoor space, a trivial solution would be to deploy sensors at every possible corner of the indoor space, collect data continuously, and infer the current conditions at different parts of the space. However, there are two major problems with this. Firstly, even if the sensor devices become cheaper, deploying them in large numbers still incurs a significant cost for purchasing, deployment, and management. Secondly, for certain locations the permanent installation of physical sensors is infeasible. For example, head-level temperature sensors would be inconvenient for residents. Though using energy harvesting and parsimonious scheduling reduces the maintenance cost of changing batteries, it neither solves the problem of long-term deployment of large numbers of sensors nor the problems of tackling sensor failures or that some sensor positions are not acceptable for building residents. Thus, a different approach is required that can provide fine-grained, continuous sensor measurement without the cost and management burden of large permanent sensor deployment.

Approach. In this article, we aim to reduce the costs of sensor management by mixing real sensors with virtual sensors, where virtual sensors are models that can use the readings of one sensor to predict sensed data at another location where there is no physical sensor present. The obvious question is how to create the virtual sensors and how reliable are they? Our approach comprises a short training period and a long term operational period. During the training period, temporary sensors are deployed in the building and measurements are collected. This data is used to train virtual sensors that predict the readings of a target sensor using the readings of the predictor sensor. The selection of predictor sensors is data-driven in that the predictor sensors are not necessarily near the target sensor and may not even sense the same phenomena.

After the training period, the whole system is analyzed to determine an optimal mix of physical and virtual sensors to achieve the purpose of the sensor network. Sensor selection is a multi-objective optimization problem balancing the accuracy and robustness of the predictions, with the cost of deployment. During the operational period, the selected deployed and virtual sensors are used to deliver data for the application.

The main problem addressed in this article is how to select an optimal set of physical and virtual sensors for long term monitoring so that the total cost of the deployed sensors is minimized and the reported values of the sensors and the fault-tolerance for those predictions are within an acceptable error bound for the domain application. This is particularly important for companies who are selling **sensing as a service**. They do not charge their clients based on the number of devices, but rather for the knowledge gained from the gathered data. If they can meet the application requirement with minimal fixed hardware that increases their revenue. The main contributions of this article are summarized in the following.

- We propose a new algorithm for measuring a 3D space using an optimal mix of physical and virtual sensors for a building sensor network. The algorithm selects physical sensors for cost, fault-tolerance, and accuracy.
- We evaluate the algorithm using real-world building networks and a vineyard soil monitoring network, demonstrating that there are opportunities for significantly reducing the complexity of sensor network deployments and the ongoing costs of their maintenance.
- We show that over a long time period of more than a year our algorithm is able to maintain accurate predictions and to tolerate sensor failures using only one month of training data.

2 BACKGROUND

Many studies have addressed the problem of how to achieve energy-efficient buildings. There are two fundamental approaches: (i) reducing electricity consumption, and (ii) constructing energy-efficient building structures. In this section, we discuss the pivotal role of sensor networks for both of these categories. Since managing a sensor network is not a trivial task, a number of approaches are used to enable a sensor network as a simple data collection tool. We briefly discuss some of these approaches along with their limitations as a building monitoring tool.

2.1 Electricity consumption reduction

HVAC and lighting systems are identified as the highest contributors to energy consumption in a building [26]. As a significant proportion of this energy consumption is attributed to inefficient usage, there is scope for significant energy saving by using efficient means to manage HVAC and lighting units. One of the prominent approaches is to identify the occupancy of a room and control the devices accordingly [3, 13].

Though most works focus on identifying occupancy without deploying a large number of sensors, doing away with any sensors is not feasible. Another approach is to reduce the over-utilization of HVAC and lighting systems. This method requires a thorough understanding of how much usage is sufficient. Personal thermal comfort deals with this question [16, 27]. Similarly, lighting controls based on preference have also been studied [14, 32]. However, this can only be achieved if sufficient data is available to learn personal comfort and preferences, and ample data is available to assess the indoor conditions continuously.

2.2 Energy-efficient buildings

Effective control of energy in legacy buildings can certainly save energy. But when constructing new buildings the opportunity exists to use novel materials and designs to minimize the energy that will be required for occupants' comfort [5]. When a new building is planned, its performance is predicted at the design stage using building physics models, often supported by simulation software. Many countries mandate energy efficiency standards for new buildings, requiring the new designs to be rated for their energy efficiency using these models [9]. However, existing models may not be accurate for novel designs. Thus, in-situ measurements are important for understanding the performance of novel designs, and ultimately for enabling effective policy on energy-efficient buildings.

2.3 WSN as a data collection tool

As sensors are an integral part of energy-efficient buildings, the goal of this article is to develop a building monitoring sensor system which is economical, easily manageable and provides accurate, fine-grained continuous data. Since the inception of wireless sensor networks (WSN), a large body of work has focused on fine-grained sensing from minimal measurements. Even though the primary objectives of these techniques overlap with each other, i.e., energy-efficiency and efficient sensor management, they differ in terms of their approach and application scenario. In the following, we briefly discuss these techniques by grouping them into four broad categories. A summary of these techniques is also provided in Table 1.

Data estimation: As data transmission consumes the most energy for an embedded sensor device, data estimation techniques are employed to reduce the amount of data transmission per sensor, thus energy consumption [11, 18, 29, 30]. Hybrid methods combine on-node temporal compression followed by spatial compression during data collection [1]. Nodes with correlated readings can form spatial clusters to reporting only one measurement [37]. Clusters can be updated

over time if the correlations change. Data estimation methods can also be used to tackle missing sensor data, node failure, or communication failure.

Coverage problem: In many wireless sensor networks, there are more nodes than the optimal requirement. The reasons can be non-overlapping of the sensing range and the transmission range, infeasibility of careful deployment, or for robustness to node failure. This leads to redundant sensor nodes in the network. The coverage problem selects a subset of active nodes (post-deployment) that are sufficient to cover the whole area while ensuring connectivity [6, 25]. The subset of active nodes is changed periodically so that there is balanced energy expenditure by the nodes.

Node scheduling: Node scheduling is similar to the coverage problem, where the data from a subset of deployed nodes are sufficient. However, here the active node selection is not only based on coverage criteria, but it can be based on spatial and/or temporal correlation among the nodes [15, 20, 34, 37]. Predictors based on correlation are also studied in our previous work [8, 31] using linear or hour of day prediction models.

Compressive sensing: This is another technique that reduces the amount of traffic within the network. If the sensor data is sparse in some domain then using the compressive sensing technique sensor data of all the sources can be reconstructed from the measurements at a few sources [23, 33, 36, 38].

Field estimation problem: For some applications, the cost of deploying sensors at all is very high, but there is still a need for fine-grained field measurements. Applications include monitoring of soil moisture [35] or air [24] or water [12] quality. Here it is assumed to be impossible to deploy sufficient sensors to cover the field at all times. Instead, some physical sensors are deployed and then a statistical model of the application field from models such as computational fluid dynamics are used to estimate optimal sensor positions. Wu *et al.* investigate how to optimize the positions of physical soil moisture sensors in order to maximize either the entropy, mutual information or to minimize the mean squared error for each virtual sensor [35]. Marjovi *et al.* consider an air quality network with mobile sensors. They compare the performance of a log-linear regression model and a deep learning framework that learns data dependencies [24]. Du *et al.* generate optimal long-term sensor placements based on wind models and knowledge of annual monsoon seasons [12]. Our approach differs because deploying temporary, fixed sensors in building applications is not too expensive. Thus we can build prediction models and determine the best positions for physical sensors based on actual measurements rather than *a priori* field model. In addition, we investigate the fault tolerance and stability of sensor placements.

2.4 Research gap analysis

It is evident that sensor-based data collection is an essential requirement for buildings to achieve energy-efficiency. However, most of the existing sensor network optimization techniques are applied at the post-deployment stage to minimize the activity of deployed sensors. Although they save energy on the nodes, they do not reduce the number of sensors required. Though there are pre-deployment strategies for selecting a limited number of sensors [12, 21, 24, 28, 35], they require applications such as outdoor environmental monitoring where a statistical model of the measured phenomena and deployment environment are known *a priori*. In this work, we focus on a pre-deployment strategy for monitoring in buildings where the statistics of the field is not well known. We develop a method for fine-grained continuous data collection using only a minimal set of post-deployment sensors. We extend previous sensor selection methods by simultaneously optimizing cost, fault-tolerance, and accuracy.

Table 1. Techniques to infer fine-grained sensor data from limited measurements.

category	general technique	limitations
data estimation	correlation in sensor data is exploited to predict one sensor value with the help of other	applicable only for short-term estimation
coverage problem	avoid overlapped-sensing by multiple sensors; use only a subset of sensors that can cover the whole area (or all the points)	*a priori* knowledge about the field is required and the statistics should hold true over time
node scheduling	select only a subset of sensors as active such that the whole monitoring region is covered avoiding overlap by multiple sensors	scheduling can be done only in short burst to cope with varying dynamics of the field
compressed sensing	exploiting sparsity in the sensed data, recover the whole dataset from very few samples	*a priori* knowledge about the field is required and the data should be sparse in certain domain
field estimation	careful sensor placement that can cover the entire sensing region or the points of interest by exploiting the distribution of the measured parameter in the field	*a priori* knowledge of physical model(s) for the sensed phenomena and the deployment environment are required

3 BUILDSENSE FRAMEWORK

In this work, we present BuildSense, a framework for monitoring spaces such as buildings and environmental applications. The main objective of the framework is to create and manage a sensor network for supporting energy-efficient buildings. Key features of BuildSense include optimal deployment strategy, low-cost and fault-tolerant deployment, ease of management, building-level customization, fine-grained and continuous data collection, robust and accurate sensor data. To attain all these features simultaneously, a number of design choices are made and accordingly algorithms are developed.

3.1 Design principles

At the core of its design, BuildSense needs to ensure that accurate, fine-grained, continuous sensor data is available. It is suitable for scenarios where the relative levels between sensor readings at nearby locations are stable even when the absolute sensed values are constantly changing. Many phenomena in building monitoring have this feature, including temperature, relative humidity, and soil moisture [35]. Additionally, its goal is to reduce the deployment and management cost of a large number of sensor devices. To fulfill these two divergent requirements, BuildSense uses an optimal mix of physical and virtual sensors, where virtual sensors use a prediction model that can predict sensor data accurately with the help of other physical sensors. BuildSense addresses the problem of predicting sensor values at a given set of monitoring locations in 3D space. BuildSense does not simultaneously address the problem of optimizing sensor locations to satisfy short-range communication constraints. Instead we assume a star topology for wireless communication where nodes can reach the sink in a single hop. This scenario is becoming common with the advent of low power, long range wireless technologies such as LoRa, Sigfox, and NBIoT. The number of physical sensors is kept as low as possible, which leads to reduced costs. In this way, the use of virtual sensors ensures fine-grain measurement with a minimal sensor infrastructure. However, solving this problem raises a number of rudimentary questions: (a) how to create a virtual sensor? (b) how many physical sensors are required to ensure sufficient granularity in the data? (c) how to ensure

the accuracy of the reported data over a long time period of months or years? BuildSense follows a sense-learn-predict model that can tackle these questions.

3.2 System Model

The BuildSense framework does not assume any special system model and can be integrated into any sensor network. In other words, it inherently supports easy customization for any building or other environmental monitoring settings. The framework works in three phases: (i) data gathering, (ii) training, and (iii) operation, which not only allows building-level customization but also helps to tackle the rudimentary questions mentioned before. Fig. 1 shows how the three phases constitute the framework.

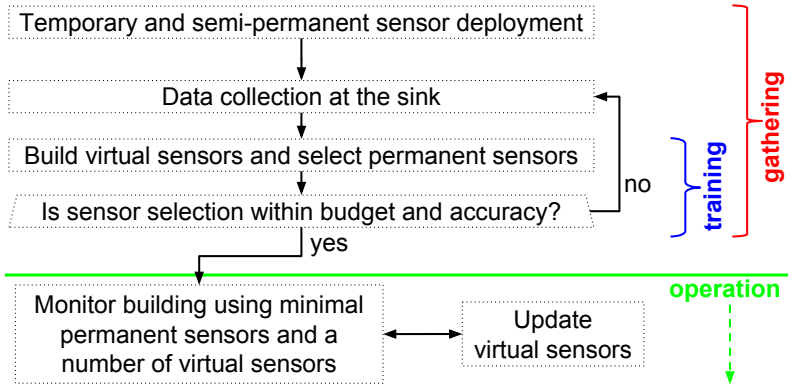

Fig. 1. The three phases of the BuildSense framework.

During the **gathering phase**, a large number of sensors are deployed in the building. This temporary network collects data for a period of a few weeks or months where all the sensors report their values to a central node, called the sink node. A centralized algorithm is used for training because it offers higher accuracy than a distributed algorithm and is appropriate for this one-off, off-line phase. A distributed algorithm for training would be an approximation to the centralized results. Also, a distributed algorithm can increase communication overhead in the battery operated nodes. For the use cases addressed in this article (training deployment followed by permanent deployment), we believe a centralized algorithm is most appropriate since the entire node set is part of the same deployment. However, in a slightly different scenario, when the deployment in a large building or region is done by multiple parties and then they collaborate to achieve energy-efficient sensing, selection of active nodes can be done distributedly. Since there is no single central node in such a scenario, a distributed mechanism is more suitable. For example, a leader election mechanism can be employed along with the centralized sensor selection mechanism but this is out of the scope of this article.

This dense sensor deployment during the data gathering phase helps to understand the characteristics of the building and helps to ensure long term sensor data accuracy using minimal infrastructure. As mentioned earlier, the goal of BuildSense is to keep the number of permanent sensors as low as possible without violating the other constraints. As a result, a large number of sensors are removed after the data gathering phase and only a few remaining sensors become the permanent infrastructure.

The **training phase** commences either after sufficient data has been gathered or the data gathering and training phases can overlap (Fig. 1). Various learning operations are performed

during the training phase by utilizing the gathered data (details are in Section 4). When the learning functions attain a predefined level of confidence, the training phase, as well as the data gathering phase, are finished. Three tasks are performed during the training phase.

T1. The first task is to learn prediction models for the virtual sensors in which the readings of a physical sensor are used to predict the readings of a virtual sensor. Sensor models are learned offline.

T2. The second task is to select the best sensor types and positions for long term measurement, to achieve high prediction accuracy at a low infrastructure cost.

T3. The third task is to control the selection of physical sensors to optimize the fault-tolerance of the system against possible sensor failure.

Once the training phase finishes, the **operational phase** kicks in. The temporary sensors are removed; the rest become the permanent physical sensors. Selection of permanent sensors is discussed in detail in the next section. During operation, measurement commences using the physical (permanent) sensors and virtual sensor models are used to predict virtual sensor values.

4 LEARNING IN BUILDSENSE

BuildSense has three learning steps corresponding to each of the tasks. This section describes the algorithms for each of the learning steps as mentioned in Section 3. Before jumping into the algorithms, let us formalize the problem statement.

4.1 Problem Statement

Suppose, we have a set of sensed phenomena $S = \{s_1, \ldots, s_n\}$, each a time-series of observations so that $s_i(t) \in \mathbb{R}$ for timestamp t. S also represents the set of sensor nodes that generate the time series (sensed) data. If a sensor s_i can be estimated using s_j during a time period T by a predictor function p_{ij} within some error bound ϵ, so that $\forall t \in T$, $s_i(t) = p_{ij}(s_j(t)) \pm \epsilon$ then we say that s_i and s_j are ϵ-neighbors written $s_i \approx s_j$ on ϵ, T. By convention, s_i is not considered a member of its own neighborhood. BuildSense needs to be fault tolerant to allow for a possible failure of a predictor sensor and to increase confidence in the predictions. We use a *k-coverage* approach for fault tolerance in which each virtual sensor is required to have at least k physical sensors that are ϵ-neighbors. Each sensor also has a cost c_i reflecting its purchase, installation, and maintenance costs. More simply, costs can be binary for each sensor, i.e., whether to deploy permanently or not. There is an upper bound, *maxCost*, on the total cost of permanent sensors for the system.

The problem we address is the following. Given training data from sensors $S = \{s_1, \ldots, s_n\}$, find a set $P \subseteq S$ of **permanent sensors** and $V = S \setminus P$ of **virtual sensors** so that

(1) There is a sufficiently **accurate** prediction function for every virtual sensor based on a training period T, i.e.,

$$\forall s_i \in V,\ \exists\, s_j \in P.\ s_i \approx s_j \text{ on } \epsilon, T.$$

(2) Predictions are **fault tolerant** in that every virtual sensor has at least k physical sensor predictors, i.e.,

$$\forall s_i \in V,\ \exists\, n_i \subset P.\ size(n_i) \geq k\ \wedge\ \forall\, s_j \in n_i.\ s_i \approx s_j \text{ on } \epsilon, T.$$

(3) The total **cost** of permanent sensors is bounded, i.e.,

$$\Sigma_{s_j \in P}\, c_j\ \leq\ maxCost.$$

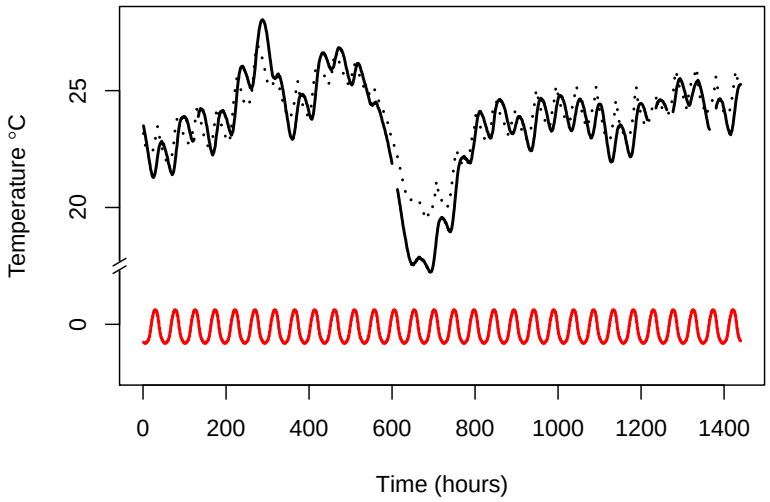

Fig. 2. Sensor readings from a bedroom (dotted) and embedded in a wall (black) and their average difference by the hour of day (red).

4.2 Sensor Predictors

The first task of virtual sensing is to learn predictors for the virtual sensors. Algorithm 1 gives the pseudo code for this process. Given a field of sensors $S = \{s_1, \ldots, s_n\}$ that have gathered data during a time period T, the goal is to learn predictor functions for each pair of sensors s_i, s_j.

In this work, we use one-to-one predictors where a single sensor is used to predict the values of another sensor. For fault tolerance, multiple, independent, one-to-one prediction functions are used to estimate of the same virtual sensor. This diversity of predictors provides protection against faulty sensors and for recognizing the need for retraining. An alternative approach could use many-to-one predictors that take multiple sensors as input to predict one virtual sensor. We chose the one-to-one strategy because many-to-one predictors do not have the fault tolerance benefits of independent predictors and since past work was inconclusive about the accuracy improvement of many-to-one predictors over one-to-one predictors [30].

Four one to one predictor functions were investigated in detail in [7]: simple nearest neighbor (use the values of the nearest valued sensor), linear regression, a cubic using the time of day [17], and the nearest hour of day neighbor. Of these, the nearest hour of day predictor had the best performance. The nearest hour of day predictor uses an offset o_h for each hour of day $h \in \{0, 1, \ldots, 23\}$ defined by $o_h = mean\{s_i(t) - s_j(t) \mid t.h = h\}$ where $t.h$ is the hour of day at time t. The predictor function for s_i is given by $p_{ij}(t) =_{def} \ s_j(t) + o_{t.h}$ and we write $s_j + o$ for this function.

Fig. 2 illustrates how the nearest hour of day prediction works. The black lines (dotted and solid at the top of the graph) show the readings for two sensors, s_1 and s_2, during a training period: s_1 is in a bedroom and s_2 is embedded in the wall of a different room. The solid red line in the lower part of the figure shows the learned hour of day offset function o for this sensor pair. This offset function has an amplitude of 1.4 ° C. It minimizes the residuals $s_1 - (s_2 + o)$.

The simple hour of day estimator turns out to be surprisingly general and effective. For the example in Fig. 2, which includes some strong context changes in the weather, the root mean

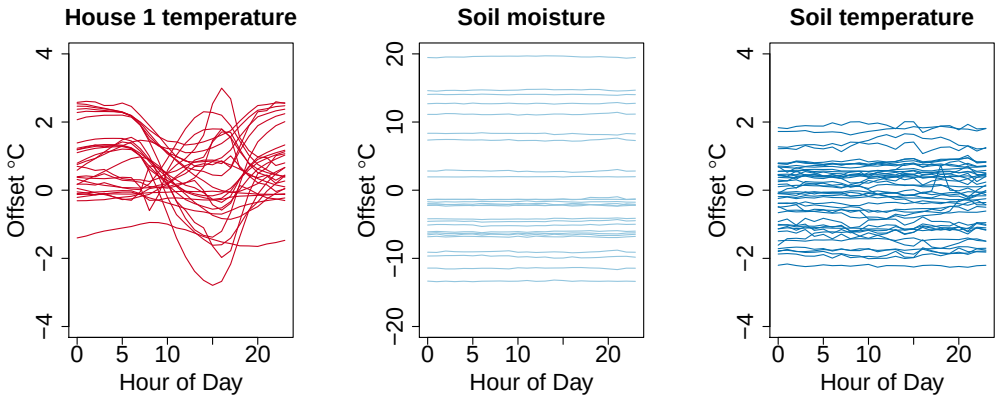

Fig. 3. Examples of offset vectors learned as nearest hour of day predictors for different sensed phenomena.

squared error (RMSE) for $s_1 - (s_2 + o)$ is 1.15° C over one month. This is sufficiently accurate for applications such as energy rating or HVAC control. Phenomena such as relative humidity can also be predicted, with RMSE of 2% to 4% [7]. The hour of day estimator can be used for many different types of sensed phenomena. Fig. 3 shows the offset functions learnt for three contrasting types of sensed phenomena and multiple sensor pairs. For building temperatures (left), the relationships between sensors are strongly dependent on the hour of day, and they vary in shape for different pairs of sensors. For soil moisture, on the other hand, Wu et al [35] observed that the relative values of observations at different spatial locations are very stable, even if their absolute values are constantly changing. The offset functions shown in Fig. 3 (middle) support that observation. Soil moisture offsets between sensors have a large range, but little dependence on the hour of day. Soil temperature (right) is slightly dependent on the hour of day, but much less so than the house temperatures.

The BuildSense algorithm for learning hour of day predictors is shown in Algorithm 1 which calculates hour of day offsets and RMSEs for every possible pair of sensors s_i and s_j. First, the residuals between sensors are used to calculate the hour of day offset function, giving a virtual sensor estimator for s_i based on s_j. The error statistic for this predictor is a root-mean-squared error (RMSE). These estimation errors are recorded in matrix E so they can be used in the sensor selection phase to optimize the selection of virtual sensors. The learned hour of day offset function for a sensor pair is recorded in matrix F. Since this predictor is additive, the offset for sensor s_i using s_j is simply -1 times the offset for s_j using s_i.

4.3 Sensor Selection

During the sensor selection phase, three criteria are considered: accuracy, fault tolerance and the cost of deployment. Algorithm 2 gives the pseudo code for the selection process. It chooses a set of permanent sensors $P \subset S$ so that all the virtual sensors $V = S \setminus P$ can be estimated with acceptable accuracy, fault tolerance, and overall cost.

The sensor selection algorithm works as follows. The error matrix E (created in Algorithm 1) is used to identify the ϵ-neighbors of each sensor. If a particular node is selected as a permanent sensor, then the values of any of its ϵ-neighbors can be predicted with high accuracy. This can be viewed as this permanent sensor *covering* a number of virtual sensors. Naturally, there are n subsets in this neighbor matrix where each one is associated with a particular node. Then a minimal

ALGORITHM 1: Learn a predictor for each sensor pair.

Input : $S = \{s_1, \ldots, s_n\}, T$ set of sensors and training period
Output : E matrix of prediction errors for each sensor pair
F prediction function parameters for each sensor pair
Variables : d differences between sensor readings
$o = \langle o_0 \ldots o_{23} \rangle$ where $o_h = mean\{s_i(t) - s_j(t) \mid t.hr = h\}$ vector of learned hourly offsets
v virtual sensor to be predicted
r residuals between actual and predicted values

for $i \in 1 : n$ **do**
  for $j \in (i+1) : n$ **do**
    $d(T) = s_i(T) - s_j(T)$; *// Residuals between sensor series*
    $o = \langle h_0 \ldots h_{23} \rangle$ from d ; *// Median hour of day offsets from d*
    $v(T) = s_j(T) + o$; *// Estimated virtual sensor values*
    $r(T) = v(T) - s_i(T)$
    $E[i,j] = RMSE(r(T))$; $E[j,i] = E[i,j]$; *// Record RMSE error*
    $F[i,j] = o$; $F[j,i] = -1 \times o$; *// Record hour of day predictor*
  end
end
return E, F

selection of permanent sensors can be achieved by reducing this problem to the minimum set cover problem. The goal of BuildSense is to select a subset of permanent nodes that cover all the sensing points and minimize the deployment cost. To increase the reliability of the system, sensing points should have multiple (k) covers. This amounts to the weighted set multi-cover problem [4]. It is well known that no polynomial time algorithm can do better than $\log n$ approximation even for unit demands and weights. Therefore we extend the greedy set cover algorithm so that it becomes also fault tolerant and cost-effective. An improved lower bound for the weighted multi-cover problem is possible in some geometric cases [4] but for the use-case scenarios of BuildSense, the correlation among the sensor nodes may not have the required shallow cell complexity geometric property and so the benefit of this specialized algorithm cannot be utilized in general.

In the baseline greedy set cover algorithm, a set is selected which covers the maximum number of uncovered elements of the universe. In this case, the equivalent criteria is that a node is selected as physical that has maximum neighbors that can be virtual. In the baseline case, the required fault tolerance for all nodes is 1. Thus, if a node is covered at least once, its weight becomes zero in the other subsets where it belongs. However, to increase fault tolerance, we can specify k_i as the required level of fault-tolerance for sensor s_i where $\forall i \in 1 \ldots n.\ 1 \leq k_i \leq size(\epsilon\text{-neighbors}(s_i))$. When the same coverage weight is used for all sensors we simply write k. Notice that k_i cannot exceed the size of ϵ-neighborhood of s_i since that gives the maximum number of possible predictors. Also s_i can always achieve coverage of at least 1 by being included in the P set, even when its neighborhood is empty.

During the sensor selection process the number of predictors still required to cover each sensor is recorded in vector Q. Initially $q_i = k_i$ for each sensor. When a node becomes a P sensor or is covered by a sensor added to P then its corresponding Q-weights are decreased by 1. The covering process continues until Q-weight of every node is zero.

To make sensor selection cost-aware, each candidate node is assigned a selection weight recorded in vector W. The criteria for selecting physical nodes is decided using the actual cost of the physical sensor divided by the current combined Q-weight of the nodes that it covers. The function *getWeightedCost* calculates the w_i value of any node s_i. A node is best for selection if it has a low

ALGORITHM 2: Sensor selection that ensures accuracy, fault tolerance and cost efficiency.

Input : $S = (s_1, \ldots, s_n)$, $C = (c_1, \ldots, c_n)$ list of sensors and associated per-sensor cost
E, ϵ matrix of prediction errors and error threshold
$K = (k_1, \ldots, k_n)$ coverage redundancy required for each sensor
$maxCost$ maximum cost for permanent sensors

Output : $P \subseteq S$ physical sensors

Variables : $W = (w_1, \ldots w_n)$ weighted cost for each candidate physical sensor
$Q = (q_1, \ldots q_n)$ coverage still required for each sensor
$Pcost$ total cost of permanent sensors

$P = \emptyset$; $Pcost = 0$; $Q = K$

// Grow P until the required fault tolerance is achieved or $maxCost$ is exceeded

while $(max(Q) > 0$ *and* $Pcost \leq maxCost)$ **do**
  for $i \in 1 : n$ **do**
    $W[i] = getWeightedCost(s_i, P, Q, C, E, \epsilon)$; *// Update weighted costs for all sensors*
  end
  $m = arg\ min_{i \in 1:n}\ W$; *// Choose s_m with minimal weighted cost for P*
  $P = P \cup \{s_m\}$
  Pcost = Pcost + C[m]
  Q[m] = max(Q[m] - 1, 0)
  for $(j \in eNhood(s_i, E, \epsilon))$ **do**
    $Q[j] = max(Q[j] - 1, 0)$; *// Reduce q_i since s_j is now predicted by s_m*
  end
end

Function $\epsilon Nhood(s_i, E, \epsilon)$**:**
  return $(\ which(E[i,] \leq \epsilon) \setminus i\ )$; *// List all ϵ neighbors of s_i excluding itself*

Function $getWeightedCost(s_i, P, Q, C, E, \epsilon)$**:**
  if $s\_i \in P$ **then**
    Return ∞ ; *// Sensor s_i is already included in P*
  end
  if $(eNhood(s_i, E, \epsilon) = \emptyset)$ **then**
    return 0 ; *// Orphan s_i must be selected for P*
  end
  $qsum = 0$
  for $(j \in eNhood(s_i, E, \epsilon))$ **do**
    $qsum \leftarrow qsum + Q[j]$
  end
  if $(qsum > 0)$ **then**
    return $C[i]$ / $qsum$; *// Sensor s_i can add value to P*
  else
    return ∞ ; *// All neighbors of s_i are already covered*
  end

w_i value. This occurs when node s_i has a low cost (numerator) and/or a high Q-weight sum for the nodes it covers (denominator). Nodes that are already in the set of physical nodes P and nodes that have no uncovered neighbors (Q-weight sum is 0) have their w_i set to ∞ to indicate they have no added value for selection. Nodes that have no ϵ-neighbors have w_i set to 0 since these nodes cannot be predicted. Such orphan nodes must be included in the set of physical nodes. The sensor selection algorithm repeatedly selects physical nodes for P until either the algorithm terminates

successfully once all nodes have achieved k-coverage or the algorithm terminates unsuccessfully as soon as the cost of the physical nodes exceeds $maxCost$.

4.4 Optimization

We have shown how to partition a network into permanent and virtual sensors for a given error tolerance and cost limit at a minimal cost. Varying the error tolerance and the degree of fault tolerance gives rise to a large set of solutions that may be suitable for a given application. But as in many optimization problems, conflicting objectives need to be satisfied. For example, increasing the required fault tolerance of a solution usually increases the cost of the physical sensor set. For this reason, the global Pareto-optimal set for a range of solutions is considered.

In this section, we consider the multi-objective optimization problem of selecting virtual sensors based on desired characteristics for the deployment. The algorithm selects virtual sensors using thresholds for RMSE, fault-tolerance, and cost. When comparing solutions, we evaluate the achieved values for these parameters across all sensors. The following metrics are considered:

Average error The average RMSE error for the virtual sensor predictions (must be $\leq \epsilon$).
Average fault tolerance The average number of physical predictors for each virtual sensor (must be $\geq \Sigma_i k_i$).
Total cost The total cost of the physical sensor deployment (must be $\leq maxCost$).

A set of solutions partitioning P and V is generated using different values for ϵ (the upper error bound), k_i (the number of covering sensors for fault tolerance) and $maxCost$ (the maximum sensor cost of the deployment). Each solution is characterized by a 3-tuple of results (a, f, c) giving the values of the three performance metrics: average accuracy, average fault-tolerance, and total cost. Let r_i be a 3-tuple solution of results from the set of all results $\{r_1, \ldots r_m\}$. r_i is dominated by some $r_j \neq r_i$ if and only if $\forall\ b \in (a, f, c).\ r_i.b \preceq r_j.b\ \wedge\ \exists\ b \in (a, f, c).\ r_i.b \prec r_j.b$ where $\prec$ means "is better than" and $\preceq$ means "is better than or equal to". For the total cost and average accuracy metrics, lower values are better, while for average fault tolerance higher values are better. Among a set of solutions R, the non-dominated set of solutions R^* are those that are not dominated by any member of the set R. The non-dominated set of the entire feasible search space is called the globally Pareto-optimal set.

Fig. 4 shows the Pareto fronts for two data sets: temperature and humidity in a rammed earth house and soil moisture in a vineyard. Solutions were generated with different parameters: $\epsilon \in \{0.5, 1, 1.5, \ldots to5\}$, $k \in \{1, 2, 3\}$ and maximum cost set to 75% of the number of sensors. Algorithm 2 was used to search for a solution for each the 30 possible parameter settings. Solutions were found for 22 cases for the house data set and 14 cases for the vineyard. The globally Pareto-optimal results for each dataset are indicated by the highlighted points. For the rammed earth house (H1), the lowest total cost (2 permanent sensors) has low fault tolerance but a relatively high average RMSE. The highest cost (8 permanent sensors) has higher fault tolerance but the lowest average error for the virtual sensors. In between, the optimal solutions offer trade-offs among cost, fault-tolerance, and accuracy. Selecting the most suitable solution from the highlighted Pareto-optimal fronts can be decided using external information about which of the optimization factors is most important for the application. Alternatively, the median value on the Pareto front could be selected.

4.5 Sentinels

BuildSense selects permanent sensors with the goal of ensuring stable predictions over a long time period. But once the temporary sensors have been removed, how does the user know whether predictions remain accurate? We propose **sentinels** for tracking prediction accuracy given possible changes in environmental dynamics during long term operation.

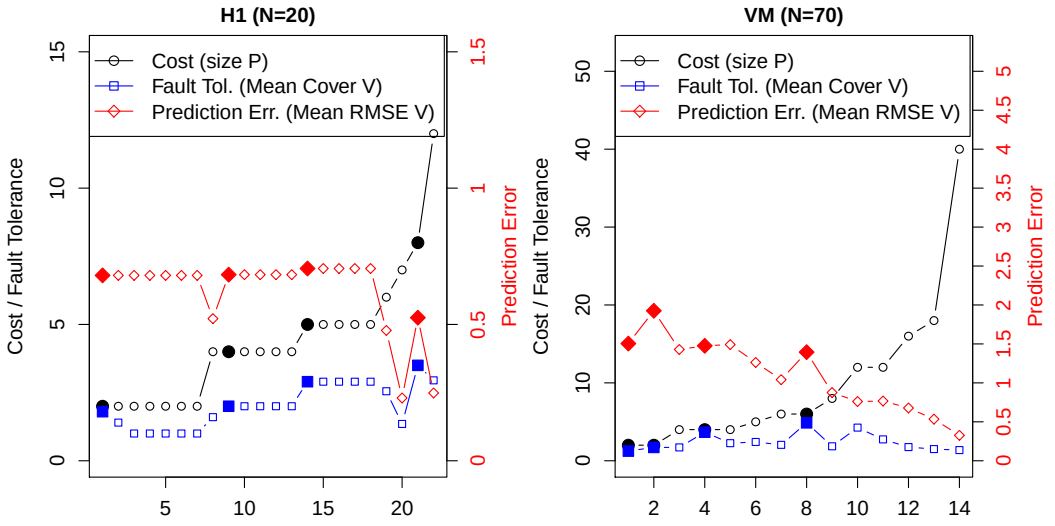

Fig. 4. Examples of Pareto-fronts (highlighted points) to minimize cost and prediction error and maximize fault-tolerance.

A sentinel sensor pair $\{s_1, s_2\} \subseteq P$ is any pair of permanent sensors that are ϵ neighbors. For these sensors, we can track the ground truth measured value as well as the value predicted by the function learned during the training period. The difference between ground truth and prediction can be calculated for every measurement point, e.g. every 30 minutes for the house data sets. During long term operation we observed three categories of differences:

Normal differences are those that are sufficiently close to indicate that the underlying environmental dynamics are stable.

Biased differences are consistently suggesting a temporary or long term change in the environmental dynamics between the sentinel sensors.

Anomalous differences suggest a large scale environmental disturbance between the sentinel sensor locations.

During operation, the differences between measured and predicted values for sentinel pairs are recorded. If the error for one or more sentinel pairs drifts into a biased or anomalous category too many times then the user can be warned. In the case of biased readings, the user may decide to trigger the retraining phase with existing data to learn new predictors. Additionally, some temporary sensors may be redeployed and further data gathered before the system is retrained. On the other hand, anomalous differences are most likely caused by ad hoc user behaviors of the occupants. For example, occupants may use small electric heaters on some cold days. Retraining is not a good strategy for anomalous readings since the changes are short term and unpredictable. But sentinel anomalies can be used to alert the user of low confidence for predictions across certain building boundaries. The value of using sentinel pairs to track the proportion of anomalous, biased and normal predictions is evaluated in Section 5.5.

5 EVALUATION

The goal of BuildSense is to set up an easily manageable sensor infrastructure for any energy-efficient building or other sensor field requiring fine-grained, long term monitoring. It advocates

Table 2. Sensor network datasets

ID	Application and sensor type	Sensors	Months	Training Month
HH	Houses relative humidity	7	22	December
HT	Houses temperature	36	22	October
H1	House 1 (insulated rammed earth)	18	22	October
H2	House 2 (monolithic rammed earth)	20	22	November
VM	Vineyard soil moisture	70	1	July
VT	Vineyard temperature	91	1	July
VA	Vineyard all sensors	161	1	July

for a sensor network with minimal sensor deployment, yet supports fine-grained data acquisition with high accuracy. To validate the framework, we address the following questions using the data collected from two real-world building monitoring sensor networks and a soil monitoring sensor network for a vineyard.

Feasibility: How feasible is virtual sensing in real-world applications?

Cost: How effective is the algorithm at finding optimal low-cost monitoring solutions? What levels of cost saving are possible?

Prediction Accuracy: How accurate and robust are the virtual sensor predictors?

Prediction Stability: Are accurate predictions able to be maintained for long operational periods?

5.1 Data Sets

Before evaluating BuildSense with respect to these performance criteria, let us first describe briefly the datasets that are used for the performance measurements. BuildSense is evaluated using data from two sensor networks that monitor energy-efficient rammed earth houses in the hot-arid climate of Kalgoorlie in Western Australia's goldfields [5]. Non-personal data has been made publicly available at https://doi.org/10.26182/5bff840f65169.

The purpose of the house monitoring networks is to investigate to what extent adopting passive solar design principles can reduce dependence on artificial climate control. The two single-floor houses have identical design and orientation, but one was built with the traditional solid rammed-earth walls (**monolithic**) and the other with an insulating polystyrene core into the walls (**insulated**). The heterogeneous monitoring networks include commercial sensors for temperature and relative humidity, bespoke sensors including temperature sensor profilers embedded in the walls of the buildings, temporary sensors that could be deployed only while the buildings were unoccupied, a local weather station and public data from the nearest Bureau of Meteorology weather station. The sensing intervals range from 5 minutes to 30 minutes. Linear interpolation was used to fill in up to 4 hours of missing values where necessary. Half hourly observations were then selected from each sensor stream, giving around 1440 observations per sensor per month.

Fig. 5 shows the floor plan and the placement of sensors in each of the houses, where KIT, LIV, BW, BS, BE refer to the Kitchen, Living Room, and Bedrooms West, South and East, respectively. The WASHING area comprises a toilet, bathroom, and laundry. The environs of the houses are instrumented with over 150 sensors, mixing temporary and long-term sensing.For each house, as shown in Fig. 5, M1 to M4 each indicates profilers of 8 temperature sensors embedded within the rammed earth walls, and H1 to H6 indicate additional temperature sensors embedded in the walls. A1 to A5 are temperature and humidity sensors at ceiling height. In addition, A1 to A5 indicate the positions of 20 temporary head-level temperature and humidity sensors that were deployed before

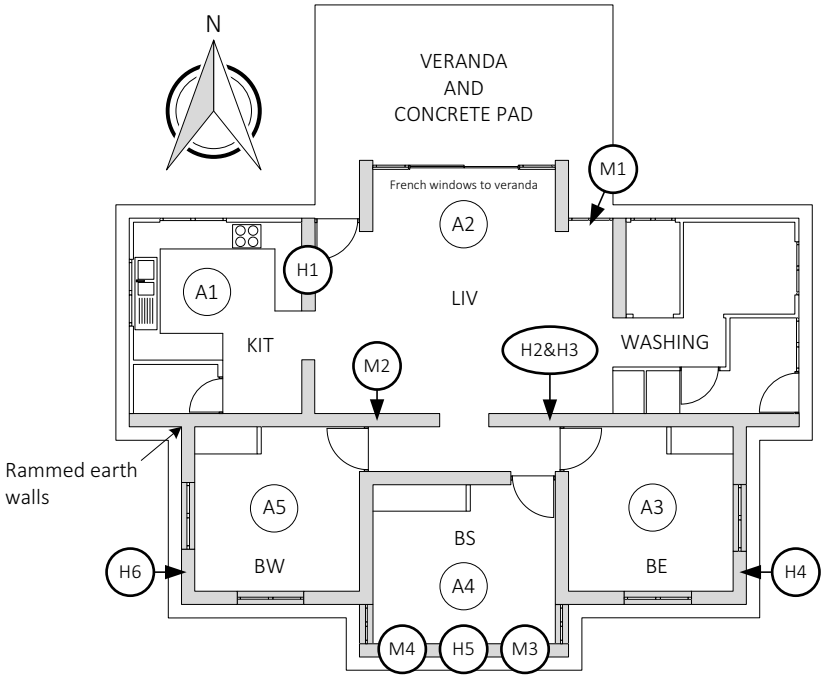

Fig. 5. House plan and sensor placements for both the monolithic and insulated houses. See the text for an explanation of the sensor labels.

the building was occupied. Moreover, a weather station with 5 sensors is placed between the two houses. The monitoring sensor networks ran continuously for 22 months.

Additionally, we evaluate BuildSense using data from a German vineyard, Nierstein bei Mainz, near Frankfurt [19] measuring soil moisture (volumetric water content %) and soil temperature at soil depths of 15cm, 30cm and 70cm at 20 locations. Data is available from http://wsn.csse.uwa.edu.au/. After removing the incomplete time series, there are 161 sensor time series in total. For evaluation purposes, we partitioned the data into periods of one month. Table 2 summarises the datasets used for evaluation.

5.2 Feasibility

In this section, we evaluate how practical virtual sensing is in several different real-world scenarios by investigating the number of potential physical sensors available to predict any virtual sensor. Fig. 6 shows the distribution of the sizes of ϵ-neighborhoods for seven evaluation datasets (see Table 2). The neighborhoods for each sensor are evaluated using three error bounds: $\epsilon = 0.5, \epsilon = 1, \epsilon = 5$. The figure classifies sensors by the size of their neighborhoods using five classes: > 10 neighbors, 5 to 9, 2 to 4, 1 or 0 neighbors (that is, sensors with no predictors). The more nodes that have large ϵ-neighborhoods, the greater the potential gains from virtual sensing.

Fig. 6 demonstrates that, as expected, as epsilon increases (across each group of 3 in the figure) so the proportion of nodes with many neighbors increases. Even for error bound as low as 0.5, virtual sensing has good potential because the majority of sensors have at least 3 ϵ-neighbors (dark shading). The exception to this rule is relative humidity sensors where virtual sensing is only feasible for $\epsilon = 5\%$ or higher.

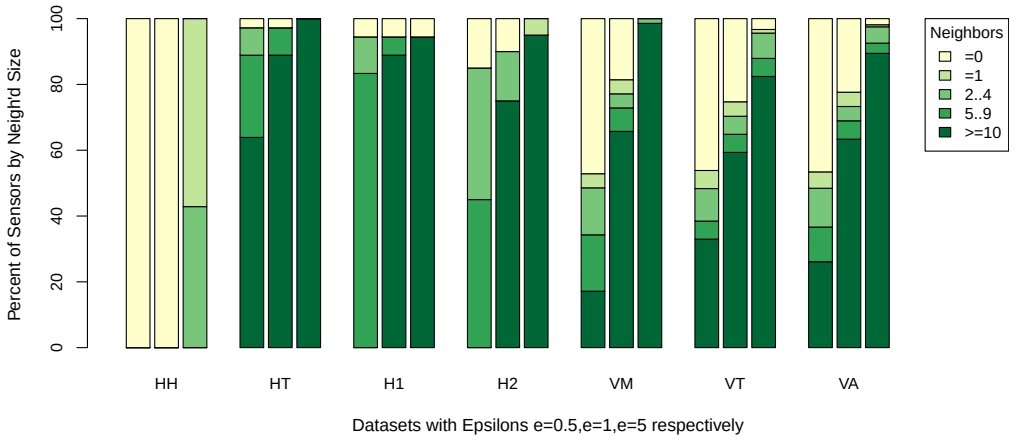

Fig. 6. Distribution of ϵ-neighborhood sizes for $\epsilon \in \{0.5, 1, 2\}$ (respectively) for contrasting datasets.

Virtual sensing is feasible for different types of sensor data, not just smooth temperature curves but also more volatile phenomena such as relative humidity and soil moisture. It is of note that neighborhoods containing more than 10 predictors (darkest green in the figures) are common. This shows that our nearest neighbor predictor model offers much greater scope for virtual sensing than the disk sensing range model used in previous studies. In the disk model, each sensor's predictor neighborhood contains only those sensors within a limited physical distance. But the nearest hour-of-day neighbor approach can find neighbors anywhere in the sensor network, even between sensors of different phenomena.

5.3 Cost-aware sensor selection

As mentioned earlier, if all the sensors are homogeneous in terms of cost, then minimum sensor selection ensures minimal deployment cost. However, if the sensor nodes have various costs then a different selection of the same number of sensors can yield a lower deployment cost. The dataset as described in Section 5.1 is collected using three different types of sensors with the cost of 60, 83, and 100 *AUD*. Clearly, if all the selected permanent sensors cost 100 *AUD* each as opposed to 60 *AUD* each, the cost of the deployment would be higher. Algorithm 2 provides a cost-aware sensor selection mechanism. Table 3 shows a comparison between this cost-aware sensor selection and a simple sensor selection where costs are not considered while selecting permanent sensors. To show how close these solutions are as compared to an optimal solution, we find the optimal solution (the possible least cost deployment) by using a linear programming formulation.

The results summarized in Table 3 and Fig. 7 are generated based on a set of simulation experiments. In particular, the results for simple coverage and cost-aware coverage are generated (for particular parameter set) by implementing the algorithm using Python. On the other hand, the linear programming formulation to find the optimal solution is solved using the *lp-solve* tool.

The cost of deployment depends on the number of permanent nodes, which is dictated by the accuracy requirement. The error threshold to decide the ϵ-neighbors help in this selection. Similarly, the k-coverage requirement to introduce redundancy also increases the number of permanent nodes; thus the cost of deployment. The results in Table 3 show that cost-aware selection achieve a

Table 3. Cost comparison of the permanent sensor deployment (in *AUD*) using various sensor selection methods.

coverage (k)	error threshold (ϵ)	optimal coverage	simple coverage	cost-aware coverage
1	0.5	392	480	392
1	1.0	196	240	196
2	0.5	722	1172	872
2	1.0	316	526	392

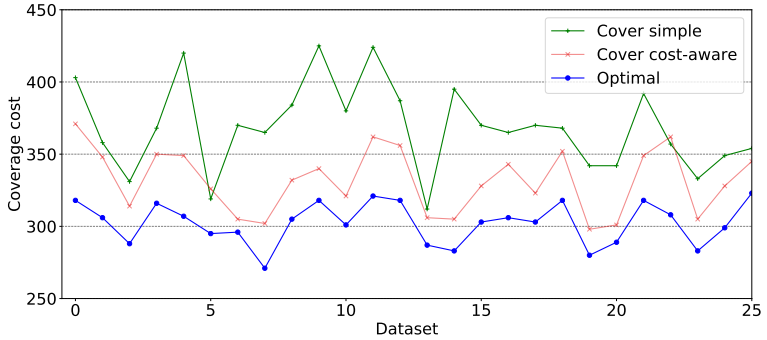

Fig. 7. Coverage cost of the permanent sensor deployment based on the three sensor selection algorithms, where coverage cost is equivalent to the total purchasing cost of the deployed nodes in *AUD*.

lower cost coverage than the simple selection mechanism and it is close to the optimal value. Since optimal selection by the linear programming formulation is not feasible for large numbers of nodes, our algorithm provides the best feasible solution.

Cost-aware sensor selection uses a greedy heuristic that neither provides the optimal sensor selection nor does it provide the guarantee of improved sensor selection as compared to the simple cover algorithm. However, statistically, it provides a better sensor selection in terms of the cost of permanent deployment. This claim is substantiated in Fig. 7. The same deployment of sensors as in Table 3 are considered with varying correlation properties. We use 25 different datasets all with the same cost and $k = 1$ but the ϵ-neighbors are significantly different from each other in each of these datasets. In most of the cases, cost-aware sensor selection is much more cost efficient than the simple sensor selection mechanism and close to the optimal result. The variation in the dataset based on the correlation among the sensor nodes signifies that the cost-aware sensor selection is not biased towards one particular type of node correlation. Hence, it showcases a general improvement for sensor selection.

5.4 Prediction Accuracy

BuildSense finds virtual sensors using prediction models that replace physical sensors. But how accurate are those prediction models? And how robust? The prediction model used in BuildSense is a simple regression model in which the final virtual sensor readings are calculated by adding a per-hour offset to the measured value. The hourly offsets used for virtual sensors are determined by a least squares approach that fits a model to the training data. Root mean squared error (RMSE) is used to define the accuracy of a particular model and for choosing the best model from available

models. This section investigates the properties of BuildSense that contribute to the quality of those predictors: their accuracy and robustness.

Consider a predictor sensor p for sensor s and a learned offset o so that the virtual sensor v is defined by $v = p + o^*$, where o^* maps the correct hourly offset at each point in the sequence. The sequence has length n. The residual r is a time series of the difference between the predicted and observed values for the virtual sensor: $r = v - s$. The quality of the virtual sensor models is assessed using the following metrics for the quality of regression models.

(1) **RMSE:** $\sqrt{\Sigma(r \times r)/n}$. The RMSE should be as small as possible.
(2) **Residual mean:** $\Sigma\, r/n$. The mean of the residuals should be as close to 0 as possible.
(3) **Homoscedasticity:** The variance around the regression line should be the same for all values of the prediction variable. For example, the variance of the residuals should be similar for both high and low sensed values. This can be measured by first partitioning the residuals into groups for each value of the prediction variable (e.g. to the nearest degree) and calculating the interquartile range (IQR) for each group. Homoscedasticity is the standard deviation of these IQRs. Its value should be as close to 0 as possible.
(4) **Auto-correlation:** represents recurring patterns in the residuals related to the time of observations. This can be calculated by grouping the residuals according to the hour of the day, calculating the interquartile range (IQR) and taking the standard deviation of the IQRs. This auto-correlation measure should be as close to 0 as possible.
(5) **Correlation:** $cor(r, p)$. This measures the correlation between the physical sensor values and the residuals from the virtual sensor. Weak correlation indicates that the virtual sensor model is independent of the underlying context, whereas a strong correlation between these measures indicates that the model may be missing some parameter. We measure correlation using the pairwise Spearman correlation coefficient which ranges from -1 to 1. Values close to 0 indicate the weak correlation we desire.

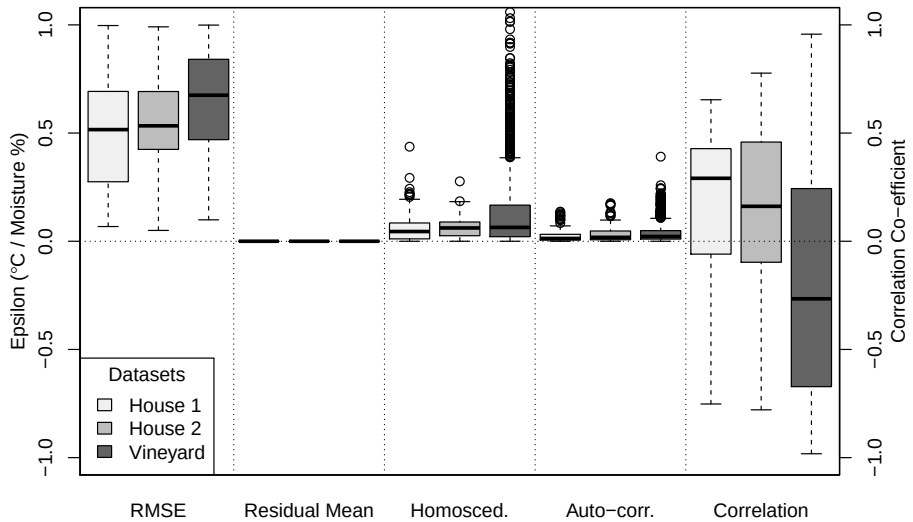

Fig. 8. Accuracy metrics for virtual-physical predictors of three datasets for $\epsilon = 1.0$.

In order to investigate the quality of BuildSense prediction models, we investigate the learned predictions for all eligible pairs of sensors from the two rammed earth houses and vineyard datasets. For each dataset, we select all virtual-physical sensor pairs with $\epsilon \leq 1.0\,C$ (or 1.0 % soil moisture). Each of these pairs has around 600 hourly readings from its training month. We calculate the predicted values of the virtual sensor and compare those with the measured values for that sensor. This gives approximately 600 residuals for each sensor pair, for which the five accuracy metrics are calculated. Fig. 8 shows the distribution of results. The units are $^\circ$C or % soil moisture for all metrics except the correlation coefficient, which is on the scale $[-1, 1]$. As expected, all estimator RMSEs are $< \epsilon$ and the residual means are near 0 since both these conditions are used in the least squares construction of the hour of day predictor. But it is notable that the median RMSE is much lower (0.25 to 0.4 degrees) than ϵ for all three datasets. The variance of the residuals (homoscedasticity) is largely independent of the prediction variables (temperature or soil moisture). The median of the correlations between predictions and the measured phenomena are also close to zero (median of -0.1596 over all pairs). However, there are some pairs where the correlation value indicates a dependence. It may be useful to use this metric to exclude such pairs from virtual sensor predictions. The autocorrelation statistic is close to 0 for all datasets which is as expected since the hour of day predictor already accounts for diurnal effects.

5.5 Stability and Retraining

BuildSense uses a sense-train-predict approach in which sensor data from a certain training period is used to train a prediction model. In this section, we evaluate the performance of BuildSense over a long time period. How much do the environmental dynamics change? Is retraining beneficial? And, if so, what should be the triggers for retraining and how often is it required?

To test the long term stability of predictions we choose pairs of sensors from the house data sets with at least 400 days of data. There are 95 such sensor pairs. For each sensor pair and training month, we compared the ground truth of the measured values with the value predicted by the trained models.

Five scenarios were considered and are reported in Figure 9. As a baseline, the system was retrained every month. In this case, the nearest hour of day estimates give the best fit for the real data. We then consider a scenario where retraining is applied after several months of operation. The system is trained in November and then retrained in February when the season and occupancies of the houses have changed. The same set of permanent and virtual prediction pairs are used each time, but the estimation model is re-learned on February data. Finally, we consider three scenarios with a single training month and no retraining during operation. The nearest hour of day model learned in the training month is applied to all other months of operation. The no-retraining scenario is tested for three contrasting training months: November (temperature range 7 to 38, median 22 degrees C), February (range 15 to 44, median 28) and April (range 7 to 27, median 17).

Figure 9 shows the achieved accuracy (RSME) for four representative sensor pairs under the five training scenarios. With reference to the sensor names of Figure 5 the selected pairs are monolithic house A5,M2 (top left), A5,A4 (top right) and insulated house A1,A2 (bottom left) all temperature and A2,A5 relative humidity (bottom right). The five training scenarios are labeled All for retraining every month, No+Fe train in November then retrain using February data, and No, Fe, Ap trained only once in November, February, and April (respectively).

As expected the baseline scenario (labeled All) in which the model is refitted every month has the best (lowest) range of RMSEs. More surprisingly, retraining the model (in November and February) does not offer much improvement over the scenarios without retraining. And the choice of training month only has a small effect on prediction accuracy. Notably, the long term accuracy of the

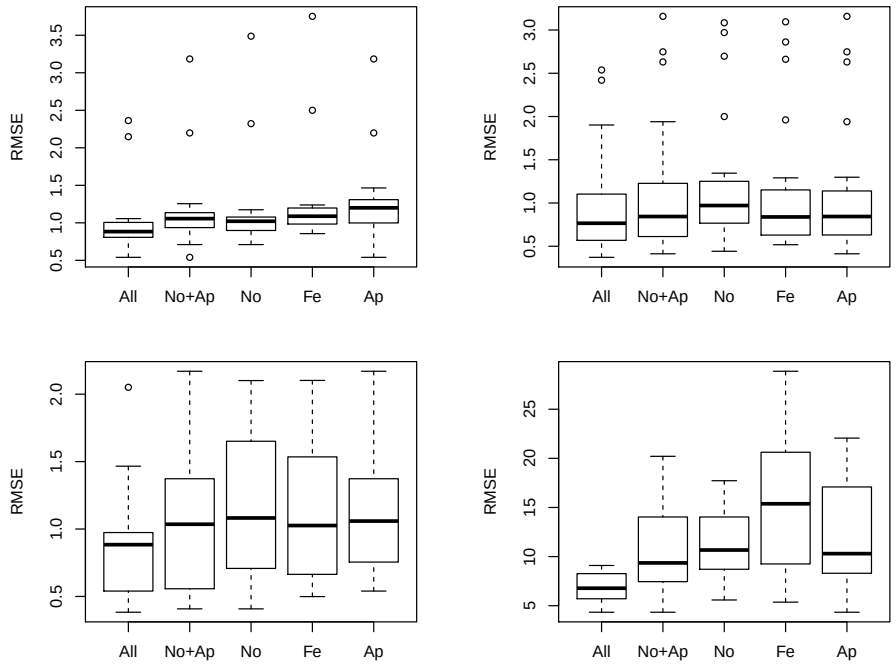

Fig. 9. Prediction accuracy for representative epsilon neighbor sensors under different retraining scenarios. See text for details.

prediction models is stable: the nearest hour of day model copes adequately with context drift connected with seasonal environmental dynamics.

We performed a second experiment to evaluate fine-grained changes in the environmental dynamics using sentinel pairs. For randomly selected sensor A3 in the insulated house (see Figure 5) we found 6 epsilon neighbors amongst the permanent sensors using $\epsilon = 0.6$. The sentinel neighbors were M2 (positions 1,2,3), A2, A5 and A4 (see Figure 5). These neighbors cover large scale differences in the building environment.

The difference between actual and predicted values was tracked for half hourly readings for each of these sentinel pairs for 12 months between November 2014 (training month) to Dec 2015 where sufficient training data was available. The ranges $[-1.5, 1.5]$ degrees Celcius were chosen as normal for house temperatures, $[-5, -1.5]$ or $[1.5, 5]$ as biased and differences > 5 or < -5 are anomalous. The results of this experiment were aggregating by months for 77 sentinel pairs and 100,935 half hourly differences. For these sentinel differences the observed categories were 1.4% anomalous differences, 14.5% biased and 84.2% normal. Interestingly none of the biased readings persisted for much more than a week so we did not see any situations where retraining was likely to be useful. Instead, we observed some periods with one-off changes in environmental dynamics. These results demonstrate that sentinels are useful for identifying such periods. Since it is not possible to predict accurately when human activities such as heating will occur we argue that the best solution is to include permanent sensors in locations likely to be affected by human changes. Expert knowledge of a building and sensor proximities can be used to review the permanent sensor set suggested by

the algorithm. The expert can then choose to include additional permanent sensors in significant locations. For example, at least one permanent sensor could be placed in each room.

6 CONCLUSION

Continuous, long-term, fine-grained, monitoring of buildings is essential for many approaches to energy-efficiency. Although deploying a large number of sensors throughout the building can provide an easy solution, it incurs high deployment and management costs. In this work, we propose BuildSense, a framework that enables continuous and fine-grained monitoring of a building with minimal sensor infrastructure. In many typical sensor networks, optimal sensor deployment may not be feasible due to the inaccessibility of the monitoring field or lack of statistical knowledge of it. On the other hand, buildings offer an accessible and well-planned environment. However, they differ significantly from each other. Thus, planning an optimal and customized (at building-level) sensor deployment becomes a non-trivial task. Though we have used the datasets from two building monitoring networks of a special kind, the data itself does not contain any special characteristics for those buildings. In other words, the dataset is sufficiently generic to provide a generic evaluation of our framework. This ensures that BuildSense can be used for any building. Customization by building-specific physical sensor selection can be applied without any customization of the algorithm. Furthermore, BuildSense is suitable for other application areas as demonstrated by our results for a vineyard soil monitoring network.

BuildSense uses a sense-learn-predict approach for customizing sensor deployments. A large number of temporary sensors are deployed during the learning phase. For the operational phase, the majority of these sensors are removed and a minimal number of sensors are selected for permanent monitoring. Using a mix of physical and virtual sensors, BuildSense ensures granular and continuous sensor data with sufficiently high accuracy, fault tolerance and low cost within a given budget. Overall, we were able to reduce the cost of a building sensor network by 60% to 80% by replacing physical sensors with virtual ones while still maintaining the accuracy of $\leq 1.0^{\circ}$ C and fault-tolerance ≥ 2 predictors per sensor. We have also demonstrated that accurate, fault-tolerant predictions are sustained for more than one year.

There are several interesting avenues for future work. One area for investigation would be to incorporate convenience and trustworthiness metrics into the selection criteria for virtual sensors. Small and unobtrusive sensors tend to be cheaper but also less accurate and reliable than more expensive sensors. Maybe using a set of small sensors could mask failures and so provide sufficiently accurate prediction at a lower cost than a single expensive sensor. New prediction models based on many-to-one rather than one-to-one sensor relations could also be investigated. Another interesting question is how to use external, contextual data as well as sensor readings in order to learn context-aware models to support more robust and accurate virtual sensor systems.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Australian Research Council and Western Australian Department of Housing under grant number LP140100375 and Universities Australia under grants UWiN - Underground Wireless Sensor Networks and ASPM - Advanced Wireless Sensor Networks for Soil Parameter Monitoring. The rammed earth building monitoring research was approved by the Human Research Ethics Office of the University of Western Australia (RA/4/1/7273).

REFERENCES

[1] Azad Ali, Abdelmajid Khelil, Neeraj Suri, and Mohammadreza Mahmudimanesh. 2015. Adaptive Hybrid Compression for Wireless Sensor Networks. *ACM Transactions on Sensor Networks* 11, 4 (2015), 1–36. https://doi.org/10.1145/2754932

[2] Bharathan Balaji, Hidetoshi Teraoka, Rajesh Gupta, and Yuvraj Agarwal. 2013. Zonepac: Zonal power estimation and control via HVAC metering and occupant feedback. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*. ACM, 1–8.

[3] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. 2013. Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 17.

[4] Nikhil Bansal and Kirk Pruhs. 2012. Weighted Geometric Set Multi-cover via Quasi-uniform Sampling. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7501 LNCS. 145–156. https://doi.org/10.1007/978-3-642-33090-2_14

[5] CTS Beckett, R Cardell-Oliver, D Ciancio, and C Huebner. 2017. Measured and simulated thermal behaviour in rammed earth houses in a hot-arid climate. Part B: Comfort. *Journal of Building Engineering* 13 (2017), 146–158.

[6] Mihaela Cardei, My T Thai, Yingshu Li, and Weili Wu. 2005. Energy-efficient target coverage in wireless sensor networks. In *INFOCOM 2005. 24th annual joint conference of the ieee computer and communications societies. proceedings ieee*, Vol. 3. IEEE, 1976–1984.

[7] Rachel Cardell-Oliver and Chayan Sarkar. 2016. Robust Sensor Data Collection over a Long Period Using Virtual Sensing. In *Proceedings of the Workshop on Time Series Analytics and Applications (TSAA '16)*. 2–7.

[8] Rachel Cardell-Oliver and Chayan Sarkar. 2017. Buildsense: Long-term, Fine-grained Building Monitoring with Minimal Sensor Infrastructure. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys '17)*. ACM, New York, NY, USA, Article 9, 10 pages. https://doi.org/10.1145/3137133.3137141

[9] D. Chen. 2016. AccuRate and the Chenath engine for residential house energy rating. https://hstar.com.au/Home/Chenath Accessed: 2017-06-07.

[10] Declan T Delaney, Gregory MP O'Hare, and Antonio G Ruzzelli. 2009. Evaluation of energy-efficiency in lighting systems using sensor networks. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. ACM, 61–66.

[11] Amol Deshpande, Carlos Guestrin, Samuel R Madden, Joseph M Hellerstein, and Wei Hong. 2004. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 588–599.

[12] Wan Du, Zikun Xing, Mo Li, Bingsheng He, Lloyd Hock Chye Chua, and Haiyan Miao. 2015. Sensor Placement and Measurement of Wind for Water Quality Studies in Urban Reservoirs. *ACM Transactions on Sensor Networks* 11, 3 (feb 2015), 1–27. https://doi.org/10.1145/2700265

[13] Varick L Erickson, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. 2011. OBSERVE: Occupancy-based system for efficient reduction of HVAC energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 258–269.

[14] Anca D Galasiu and Jennifer A Veitch. 2006. Occupant preferences and satisfaction with the luminous environment and control systems in daylit offices: a literature review. *Energy and Buildings* 38, 7 (2006), 728–742.

[15] Deepak Ganesan, Răzvan Cristescu, and Baltasar Beferull-Lozano. 2006. Power-efficient Sensor Placement and Transmission Structure for Data Gathering Under Distortion Constraints. *ACM Trans. Sen. Netw.* 2, 2 (May 2006), 155–181. https://doi.org/10.1145/1149283.1149284

[16] Ali Ghahramani, Farrokh Jazizadeh, and Burcin Becerik-Gerber. 2014. A knowledge based approach for selecting energy-aware and comfort-driven HVAC temperature set points. *Energy and Buildings* 85 (2014), 536–548.

[17] Carlos Guestrin, Peter Bodik, Romain Thibaux, Mark Paskin, and Samuel Madden. 2004. Distributed regression: an efficient framework for modeling sensor network data. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*. IEEE, 1–10.

[18] Himanshu Gupta, Vishnu Navda, Samir Das, and Vishal Chowdhary. 2008. Efficient gathering of correlated data in sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 4, 1 (2008), 4.

[19] C. Huebner, R. Cardell-Oliver, R. Becker, K. Spohrer, K. Jotter, and T. Wagenknecht. 2010. Wireless soil moisture sensor networks for environmental monitoring and irrigation. In *EGU General Assembly Conference Abstracts (EGU General Assembly Conference Abstracts)*, Vol. 12. 2539.

[20] Hongbo Jiang, Shudong Jin, and Chonggang Wang. 2011. Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 22, 6 (2011), 1064–1071.

[21] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. 2006. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*. ACM, 2–10.

[22] Andrew Krioukov, Stephen Dawson-Haggerty, Linda Lee, Omar Rehmane, and David Culler. 2011. A living laboratory study in personalized automated lighting controls. In *Proceedings of the third ACM workshop on embedded sensing systems for energy-efficiency in buildings*. ACM, 1–6.

[23] Chong Luo, Feng Wu, Jun Sun, and Chang Wen Chen. 2009. Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 145–156.

[24] Ali Marjovi, Adrian Arfire, and Alcherio Martinoli. 2017. Extending Urban Air Quality Maps Beyond the Coverage of a Mobile Sensor Network : Data Sources , Methods , and Performance Evaluation. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*. 12–23. http://dl.acm.org/citation.cfm?id=3108012

[25] S Mini, Siba K Udgata, and Samrat L Sabat. 2014. Sensor deployment and scheduling for target coverage problem in wireless sensor networks. *IEEE Sensors Journal* 14, 3 (2014), 636–644.

[26] U.S. Department of Energy (DOE). 2008. Buildings Energy Data Book. http://www.c2es.org/technology/overview/buildings. Accessed: 2017-05-22.

[27] Devika Pisharoty, Rayoung Yang, Mark W Newman, and Kamin Whitehouse. 2015. Thermocoach: Reducing home energy consumption with personalized thermostat recommendations. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM, 201–210.

[28] Maher Rebai, Hichem Snoussi, Faicel Hnaien, Lyes Khoukhi, et al. 2015. Sensor deployment optimization methods to achieve both coverage and connectivity in wireless sensor networks. *Computers & Operations Research* 59 (2015), 11–21.

[29] Silvia Santini and Kay Romer. 2006. An adaptive strategy for quality-based data reduction in wireless sensor networks. In *Proceedings of the 3rd international conference on networked sensing systems (INSS 2006)*. 29–36.

[30] Chayan Sarkar, Vijay S Rao, and R Venkatesha Prasad. 2014. No-sense: Sense with dormant sensors. In *Communications (NCC), 2014 Twentieth National Conference on*. IEEE, 1–6.

[31] Chayan Sarkar, Vijay S Rao, R Venkatesha Prasad, Sankar Narayan Das, Sudip Misra, and Athanasios Vasilakos. 2016. VSF: An Energy-Efficient Sensing Framework Using Virtual Sensors. *IEEE Sensors Journal* 16, 12 (2016), 5046–5059.

[32] Chayan Sarkar, Akshay Uttama Nambi SN, and R Venkatesha Prasad. 2016. iLTC: Achieving Individual Comfort in Shared Spaces. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. ACM.

[33] Mina Sartipi and Robert Fletcher. 2011. Energy-efficient data acquisition in wireless sensor networks using compressed sensing. In *Data Compression Conference (DCC), 2011*. IEEE, 223–232.

[34] Di Tian and Nicolas D Georganas. 2003. A node scheduling scheme for energy conservation in large wireless sensor networks. *Wireless Communications and Mobile Computing* 3, 2 (2003), 271–290.

[35] Xiaopei Wu, Mingyan Liu, and Yue Wu. 2012. In-situ Soil Moisture Sensing: Optimal Sensor Placement and Field Estimation. *ACM Trans. Sen. Netw.* 8, 4, Article 33 (Sept. 2012), 30 pages. https://doi.org/10.1145/2240116.2240122

[36] Liu Xiang, Jun Luo, and Athanasios Vasilakos. 2011. Compressed data aggregation for energy efficient wireless sensor networks. In *Sensor, mesh and ad hoc communications and networks (SECON), 2011 8th annual IEEE communications society conference on*. IEEE, 46–54.

[37] Sunhee Yoon and Cyrus Shahabi. 2007. The Clustered AGgregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Transactions on Sensor Networks* 3, 1 (2007), 3–es. https://doi.org/10.1145/1210669.1210672

[38] Xiaohan Yu and Seung Jun Baek. 2017. Energy-Efficient Collection of Sparse Data in Wireless Sensor Networks Using Sparse Random Matrices. *ACM Trans. Sen. Netw.* 13, 3, Article 22 (Aug. 2017), 36 pages. https://doi.org/10.1145/3085576

Received May 2018; revised January 2019; accepted May 2019