Multi-Sensor Calibration Planning in IoT-Enabled Smart Spaces

Qiuxi Zhu*, Francoise Sailhan†, Md Yusuf Sarwar Uddin*, Valérie Issarny‡, Nalini Venkatasubramanian*
*Department of Computer Science, University of California, Irvine
† Cedric Laboratory, Cnam Paris; ‡ MiMove team, Inria Paris

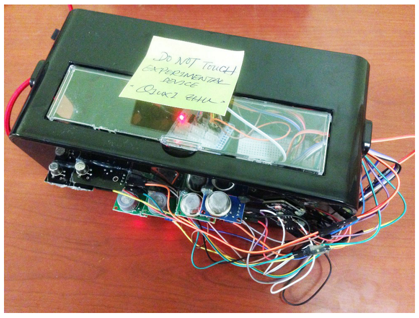
***Abstract*—Emerging applications in smart cities and communities require massive IoT deployments using sensors/actuators (things) that can enhance citizens' quality of life and public safety. However, budget constraints often lead to limited instrumentation and/or the use of low-cost sensors that are subject to drift and bias. This raises concerns of robustness and accuracy of the decisions made on uncertain data. To enable effective decision-making while fully exploiting the potential of low-cost sensors, we propose to send mobile units (e.g., trained personnel) equipped with high-quality (more expensive) and freshly-calibrated reference sensors so as to carry out calibration in the field. We design and implement an efficient cooperative approach to solve the calibration planning problem, which aims at minimizing the cost of the recurring calibration of multiple sensor types in the long-term operation. We propose a two-phase solution that consists of a sensor selection phase that minimizes the average cost of recurring calibration, and a path planning phase that minimizes the travel cost of multiple calibrators which have load constraints. We provide fast and effective heuristics for both phases. We further build a prototype that facilitates the mapping of the deployment field and provides navigation guidance to mobile calibrators. Extensive use-case-driven simulations show our proposed approach significantly reduces the average cost compared to naïve approaches: up to 30% in a moderate-sized indoor case, and higher in outdoor cases depending on the scale.**

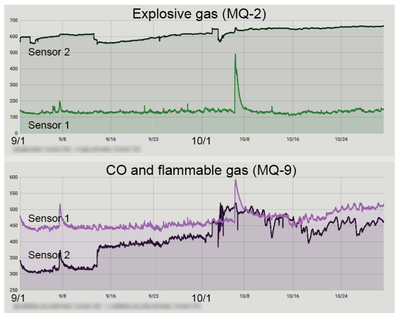
I. Introduction

The advent of IoT ecosystems with low-cost sensors and actuators is enabling the widespread deployment of smart spaces in our homes, communities, and cities. Such smart spaces are a valuable source of knowledge and can drive applications to enhance public safety, personal health and community well-being [1], [2], [3]. The associated challenges in the realization of this vision are enormous – deployment/operation cost, big data, interoperability, data analytics, and resource-efficient networking, to name a few [4], [5] – and are on the research agenda of next-generation digital sciences. In this paper, we address cost-accuracy issues that arise in the deployment of affordable IoT systems. In particular, the aggregation of relevant knowledge at scale from low-cost smart spaces is problematic since low-cost sensing solutions imply low accuracy and faster degradation/drift. Recent experiences, including by the authors, show that the relative inaccuracy of the connected devices can be alleviated through the automated and in-situ calibration of sensors – made possible due to the linearity of the bias in most cases [6]. For example, mobile platforms (including humans that carry calibration devices) can visit the smart spaces, either opportunistically or in a planned manner, to assess the biases of the deployed sensors and compensate for errors [7]. One can thus generate "sufficiently accurate" knowledge over time through the frequent calibration of the low-cost sensors in the field; however, the low deployment cost might result in increased maintenance costs! Careful planning of the calibration process is therefore essential for the cost-effective monitoring of the smart spaces – this is increasingly important as the number and size of smart spaces grow.

In this paper, we address the calibration planning problem. The aim is to develop a plan for the calibration of a large number of inexpensive (and often inaccurate) sensors in a smart space using high-integrity reference sensors that are mobile, such that (a) the deployment and operational costs for calibration are minimized while (b) maintaining a sufficient observation accuracy from the sensor measurements. More realistically, given the knowledge of a sensor's degradation characteristics, we program its calibration with respect to an observed phenomenon so as to maintain an adequate sensing accuracy while minimizing the required effort from the mobile calibrators. We formalize the above as a multi-path planning problem, which we solve via intelligent heuristics, and validate using real-world settings. Our key contributions include:

- A measurement study to help understand the calibration issue of low-cost sensors for environmental monitoring, which motivates the proposed approach for the scalable calibration of IoT-enabled smart spaces (Section II).
- Long-term multi-sensor calibration planning as a service that exploits device locality, sensor characteristics, and application needs.
- The characterization and formulation of the multi-sensor calibration planning problem (Section III) and discussion of NP-hardness.
- A two-phase iterative solution and a family of heuristic methods to enable the cost-effective planning of multi-sensor calibration in large smart spaces and over longer time periods (Section IV).
- The validation of our approach and algorithms leveraging real-world indoor and outdoor smart spaces settings from our ongoing testbeds in Irvine, CA and Paris, France; results show significant cost improvement while maintaining adequate accuracy.
- Initial steps towards a prototype of a calibration planning service for IoT smart spaces with: (i) a dashboard to map the deployment of sensor nodes over large spaces, and (ii) a mobile app that the mobile calibrators use along their calibration journeys (Section V).


(a)



(b)

Fig. 1. UCI testbed in Donald Bren Hall (Dept. of Computer Science), showing (a) a node deployed in our lab, where we reproduced the sensor deterioration process; (b) readings from two pairs of low-cost sensors on the test node, 3 months after their installment

II. Motivation and Background Experience

The widespread deployment of low-cost IoT technologies and the increasing popularity of corwdsensing enable fine grained monitoring of physical phenomena that are spatio-temporally distributed. Experiences in crowdsensing and IoT deployments indicate that, in general, measurements gathered from low-cost sensors deviate from the ground truth.

Our first set of experiences in calibration comes from the launch of an urban scale experiment with the city of Paris in 2015[1]. As part of our efforts, we developed a crowdsensing application for mobile phones for monitoring the exposure of the urban population to environmental noise pollution [8]. While crowdsensing may be a cost-effective approach for monitoring urban environmental conditions (e.g., noise) that exhibit high spatio-temporal variability, the relatively low quality of the embedded sensors as well as the uncertain sensing context required dedicated actions. We quickly learned (as anticipated) that only a low percentage of the crowdsensed measurements actually contribute to the analysis of the urban noise pollution. We thus needed to enhance the quality of the observations [9]. As a first step, we thoroughly studied the bias in smartphone noise sensing against a reference sound level meter, which helped us develop a protocol for calibrating individual handsets [10]. Through experimental studies, we demonstrated that a calibration protocol can help gather observations that can more accurately map noise pollution at the district level [11]. Unfortunately, the proposed calibration protocol places a high demand on the end-users who must actively participate in the calibration. We subsequently investigated a distributed protocol for the opportunistic multi-party calibration of devices located in the same sensing and communication range [12].

The second experience in calibration comes from lessons learned while deploying IoT-enabled environmental sensing in the SCALE project [13], [14]; here, we deployed inexpensive multi-sensor platforms (called SCALE boxes) at multiple locations worldwide including Irvine, CA; Montgomery County, MD [13], and Dhaka, Bangladesh [15], [16]. A SCALE node is a Raspberry-Pi-based multi-sensor box with middleware (SCALE client) that provides flexible interfaces for data collection for a wide range of sensor types (gas, light, air quality, temperature, seismic, camera, etc.). The SCALE client utilizes a publish-subscribe approach to organize raw data and detected events into topics that are published to a cloud (or edge) data exchange service. IoT applications subscribe to relevant topics and receive updates. The deployments allow us to obtain rich information about the sensor behaviors and especially their respective deterioration across time in various environments. In all cases, we observed a deterioration in the response of the low-cost gas sensors over time.

[1]www.inria.fr/en/centre/paris/news/launch-of-soundcity-mobile-application

At UC Irvine, SCALE was used in a smart instrumented building for everyday monitoring [17] – explosive gas sensors, useful in emergency response, were occasionally triggered externally. As part of our testbed, various sensors and monitoring devices (e.g., microphones, surveillance cameras, power meters, BLE beacons – for the location tracking of participants) were deployed in multiple floors of the CS department building. To validate our observations and generate a custom model of sensing deterioration, we developed a measurement platform to reproduce the deterioration process. Here, the custom sensor box was instrumented with 4 pairs of low-cost sensors as a "test node" (Figure 1a). Figure 1b shows the data reported by two pairs of MQ gas sensors 3–4 months after the installment: long-running MQ-2 sensors that were connected to a running node for more than two months generated different values at the same location and show significant difference in sensitivity when exposed to similar stimuli. Such difference has also been observed during/after deployment campaigns [18], [19].

State of the Art Approaches: The calibration of a sensor lies in (i) generating a controlled stimuli x and (ii) analyzing the sensor response y. Then, the true (generated) signal x, which serves as ground truth, is compared with the sensor response y so as to obtain the mapping between the sensor response and the true value; the calibration parameters are accordingly adjusted. The calibration is typically performed once in a laboratory so as to avoid the addition of any environmental interference/noise to the high-fidelity/true signal. However, the calibration does not take into account the context wherein the sensor evolves, even though environmental factors, e.g., solar radiation, strongly influence the readings provided by sensors (e.g., air temperature). Recent work focuses on automatically calibrating sensors in the field, without relying on a controlled

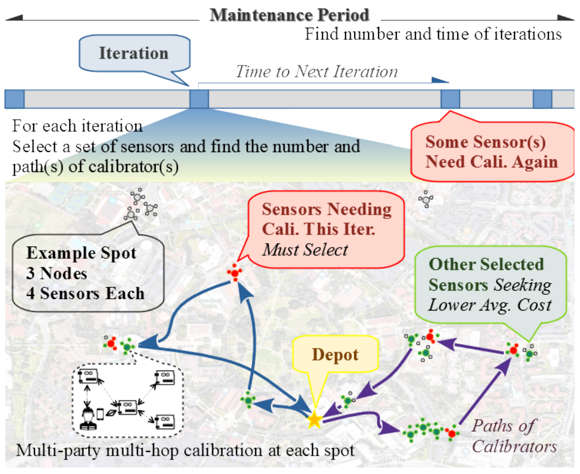

Maintenance Period
Find number and time of iterations
Iteration
Time to Next Iteration
For each iteration
Select a set of sensors and find the number and path(s) of calibrator(s)
Some Sensor(s) Need Cali. Again
Sensors Needing Cali. This Iter. Must Select
Example Spot 3 Nodes 4 Sensors Each
Other Selected Sensors Seeking Lower Avg. Cost
Depot
Paths of Calibrators
Multi-party multi-hop calibration at each spot

Fig. 2. Calibration planning over a long maintenance period.

stimuli and without a well-defined range of conditions. This is commonly referred to as *blind calibration* and it consists in calibrating a (uncalibrated) sensor using the measurements provided by a neighboring (calibrated) sensor. There are two approaches to the blind calibration of a set of sensors depending on whether the sensors are static or mobile. In the case of (static) WSNs, the common assumption is that the deployment is sufficiently dense so that neighboring sensors provide nearly identical measurements [20], [21]. Then, the inaccurate readings of an uncalibrated sensor can be compensated using the measurements provided by a nearby (calibrated) sensor. In the case of mobile sensors, multi-hop calibration [22], [23], [24] allows mobile sensors to get calibrated "indirectly" using other mobile sensors that have been calibrated by the reference sensor. Multi-party calibration [12] derives regression models to solve the calibration problem when there are multiple (more than two) participants. Recent work also investigates planning problems, e.g., [24] explores the in-situ placement of reference sensors in the field; [7] proposes a TSP based path planning algorithm for a mobile calibrator.

In contrast to the above efforts, we take a more holistic approach to low-cost calibration in smart spaces at scale. First, we consider the presence of heterogeneous sensor types with varying calibration characteristics. Our proposed approach is application-aware and is able to take into account diverse sensing needs (sensor type, sensing accuracy) presented by the context at hand. We assume the ability to utilize multiple mobile calibrators. We exploit the locality of in-situ sensors (that can calibrate each other) to reduce the cost of mobile calibrators. The multi-step approach, presented next, highlights the overall scheme for a more comprehensive and efficient calibration planning mechanism for IoT-enabled smart spaces.

III. Multi-Sensor Calibration Planning

Our objective is to enable the cost-efficient calibration planning of a given smart space with multiple types of sensors and diverse applications. The target environment assumes the in-situ deployment of a large number of inexpensive sensors with an adequate number of high-precision *mobile calibrators* (e.g., trained workers that calibrate the in-situ devices). We propose an effective approach (Figure 2) to the planning of the sensor calibration over a given – possibly long – maintenance period T: (i) First, we partition the space to form calibration *spots* by exploiting the locality of in-situ sensors; a sensor can calibrate other sensors of the same type at the same spot. (ii) Next, we carry out multiple *iterations* of calibration during the given maintenance period T. Our challenge lies in determining the number of iterations and the time at which each iteration is executed. Furthermore, in each iteration we determine: the sensors to calibrate, the number of mobile calibrators needed, and the paths taken by each mobile calibrator. The sensor calibration is then carried out accordingly over a number of Ω **iterations** so that the overall cost over T is kept to a minimum and ensures that all the sensors always comply with the *data accuracy requirements*. The planning problem then amounts to identifying the least number of iterations and planning the calibration at each iteration $\omega{=}1, 2, \ldots, \Omega$ so that the number of workers involved is also kept to a minimum. Next, we introduce the notations and assumptions underlying our formalization of the multi-sensor calibration planning problem.

A. Notations and Assumptions

A **node** n_j, $j{=}1, 2, \ldots, N$, is an IoT device that embeds one or more types of low-cost **sensors**. A **sensor type** s_k, $k{=}1, 2, \ldots, K$, refers to the capability of detecting a certain type of phenomenon (e.g., temperature, gas concentration), which usually requires a specific kind (or combination) of low-cost sensor(s). Then, we denote an individual sensor by $n_{j,k}$, and we introduce the binary **sensor presence** matrix $\mathbf{Q}_{N\times K}$ to characterize the set of available sensors so that $Q_{j,k}{=}1$ (resp. 0) if sensor type s_k is present (resp. absent) on node n_j.

1) Calibration-Related Terms: A **reference sensor** is a high-quality sensor, whose readings serve as the ground truth and therefore can be used to calibrate other sensors that sense the same phenomenon. We assume that the reference sensors are calibrated offline (e.g., in a lab before each iteration).

A **mobile calibrator** (or simply a *calibrator*) m_i is a person who carries reference sensors and visits the field to calibrate the deployed low-cost sensors. For simplicity, we assume that a calibrator can be equipped with any of the reference sensors needed across his/her journey. **Sensor calibration** takes place when a calibrator m_i visits a node n_j and stays at the spot for long enough to calibrate a sensor $n_{j,k}$.

Each sensor type s_k is associated with a **calibration time** τ_k that ranges from a few seconds to several minutes, depending on the phenomenon detected by the type of sensor and the calibration complexity. We associate each sensor type s_k with a **calibration period** T_k, which characterizes the maximum duration, during which the sensors remain valid (i.e. the measurements have sufficient accuracy) once calibrated. The period depends on the usage scenario and may be learned from empirical study.

During operation, each individual sensor $n_{j,k}$ is associated with a **time to next calibration** (TTNC) $F_{j,k}$ that indicates how soon the sensor needs to be (re-)calibrated. The TTNC matrix $\mathbf{F}_{N\times K}$ then represents the TTNCs of all the sensors.

$\mathbf{F}$ is a function of time, where each $F_{j,k}$ decreases between iterations and is reset to T_k when $n_{j,k}$ is calibrated. We further denote $\mathbf{F}[\omega_-]$ (resp. $\mathbf{F}[\omega_+]$), the matrix TTNC immediately before (resp. after) the iteration ω. Note that $\mathbf{F}$ is non-negative, i.e. $F_{j,k}[\omega_\pm]\geqslant 0$, $\forall(j,k)$, $\omega\in\mathbf{N}_+$. If node n_j does not hold a sensor of type s_k, the corresponding TTNC is infinite, i.e. $F_{j,k}[\omega_\pm]=+\infty$, $\forall\omega\in\mathbf{N}_+$, if $Q_{j,k}=0$.

A **sensor selection** is a collection of sensors (selected for calibration) represented by a binary matrix $\mathbf{\Gamma}_{N\times K}[\omega]$, where $\Gamma_{j,k}\leqslant Q_{j,k}$. If sensor $n_{j,k}$ is selected for calibration at iteration ω (i.e. $\Gamma_{j,k}[\omega]=1$), then, at the end of this iteration, its TTNC is reset to its calibration period T_k, i.e. $F_{j,k}[\omega_+]=T_k$; otherwise, its TTNC stays unchanged (during iteration ω):

$$\mathbf{F}[\omega_+] = \mathbf{\Gamma}[\omega] \circ \mathbf{T}_N + (1 - \mathbf{\Gamma}[\omega]) \circ \mathbf{F}[\omega_-] \tag{1}$$

Where $\mathbf{T}_N$ is the nodal calibration period matrix that consists of N identical rows of $[T_1, T_2, \cdots, T_K]$; "$\circ$" is the element-wise multiplication of matrices. Typically, there is no need to run an iteration if no sensor needs immediate calibration; also, special needs and unexpected changes could be easily addressed by altering the TTNC matrix. Hence, the **time to next iteration** (TTNI) after ω is the minimum TTNC of all the sensors, i.e. $t_{\omega+1}-t_\omega=\min\mathbf{F}[\omega_+]$; thus, the TTNC matrix immediately before the next iteration is $\mathbf{F}[(\omega+1)_-]=\mathbf{F}[\omega_+]-\min\mathbf{F}[\omega_+]$.

2) Smart-Space-Related Terms: We abstract a smart space as a set of spots that are such that sensor nodes deployed at a **spot** ν_l, $l=1,2,\ldots,L$, are sufficiently co-located to enable their concurrent calibration by a single calibrator. Note that this may possibly involve leveraging multi-party multi-hop calibration (Section II). We then map a smart space as a directed graph $G=(V,E)$, where each vertex $\nu_l\in V$ corresponds to a spot and each edge weight (ν_{l_1},ν_{l_2}) denotes the average time taken by a calibrator to move from ν_{l_1} to ν_{l_2}. The graph can be represented as an adjacency matrix $\mathbf{G}_{L\times L}$, where G_{l_1,l_2} is the weight on the directed edge (ν_{l_1},ν_{l_2}) (i.e. the movement cost). As long as each spot is physically accessible, G is a complete digraph and all its edges have finite weights.

The node **locations** are represented by a binary location matrix $\mathbf{D}_{N\times L}$, where each row represents a node and each column a spot. We set $D_{j,l}=1$ if node n_j is deployed at spot ν_l; or 0 otherwise. A node is deployed at a single spot, so $\sum_{l=1}^{L} D_{j,l}=1$. For simplicity, we assume that all the calibrators depart from the same spot, which is referred to as the "depot" in related work on path planning, and is conventionally indexed as the first spot, i.e. ν_1. We assume that no node is deployed at the depot. Given the node locations $\mathbf{D}$, we can derive the **spot selection** vector $\mathbf{h}_L$ from the sensor selection $\mathbf{\Gamma}$. A spot is selected for iteration ω if any sensor on any node deployed at that spot is selected in $\mathbf{\Gamma}[\omega]$ i.e.:

$$h_l[\omega] = \bigvee_{j=1}^{N} \bigvee_{k=1}^{K} \Gamma_{j,k}[\omega]\cdot D_{j,l} \tag{2}$$

In each iteration, the **path** of a calibrator is an ordered sequence that starts from the depot and visits a set of non-repeating selected spots. It can be represented as a binary matrix $\mathbf{W}_{L\times L}$, where $W_{l_1,l_2}=1$ if the calibrator visits spot ν_{l_2} immediately after visiting ν_{l_1}; or 0 otherwise. We require that all mobile calibrators return to depot in the end, i.e. $\sum_{l=1}^{L} W_{1,l}=\sum_{l=1}^{L} W_{l,1}$. The path of calibrator m_i in iteration ω is denoted $\mathbf{W}_i[\omega]$. Each selected spot is visited exactly once by one calibrator:

$$\sum_{i=1}^{M}\sum_{l=1}^{L} W_{l,l_0,i}[\omega] = h_{l_0}, \;\; l_0 = 2,3,\ldots,L \tag{3}$$

The other constraints that the matrices $\{\mathbf{W}\}$ should satisfy are discussed in Section IV-B.

B. Definition of the Calibration Cost

We consider three major types of cost: *iteration overhead*, *movement cost*, and *calibration cost*. The **iteration overhead** C_{it} is the cost of all the activities related to an iteration that are not tied to any specific calibrator, such as preparation, equipment, and the transport to the deployment, etc.

The **movement cost** C_w corresponds to the cost associated with the travel time of the calibrators while moving between spots. The **movement time** $C_{w,i}[\omega]$ of a single calibrator m_i in iteration ω can be computed from the map G and the calibrator's path $\mathbf{W}_i[\omega]$. It equals the sum of the weights on the edges between all the consecutive pairs of spots visited by the calibrator:

$$C_{w,i}[\omega] = \sum_{l_1=1}^{L}\sum_{l_2=1}^{L}\Big(W_{l_1,l_2,i}[\omega]\cdot G_{l_1,l_2}\Big) \tag{4}$$

The **calibration cost** C_c reflects the time and effort it takes to conduct sensor calibration while staying at the spots. The cost $C_{c,i}[\omega]$ of a specific calibrator m_i in iteration ω can be computed based on a given selection of sensors $\mathbf{\Gamma}[\omega]$. We indeed know the calibration protocol (and thus duration) that needs to performed per sensor type. We further assume that the calibration that happens at the same spot is done in parallel. Thus, the **calibration time** that m_i spends at spot ν_l equals the maximum τ_k of all selected sensors at that spot (i.e. $\Gamma_{j,k}=1$ and $D_{j,l}=1$). Then $C_{c,i}[\omega]$ equals the sum of the calibration times at all the spots assigned to m_i:

$$C_{c,i}[\omega] = \sum_{l=1}^{L}\left(\max_{j,k}\Big(D_{j,l}\cdot\Gamma_{j,k}[\omega]\cdot\tau_k\Big)\cdot\sum_{l'=1}^{L} W_{l,l',i}[\omega]\right) \tag{5}$$

The **work load** of any calibrator m_i at iteration ω is the sum of his/her movement and calibration time:

$$C_i[\omega] = C_{w,i}[\omega] + C_{c,i}[\omega] \tag{6}$$

For any iteration, we assume that the maximum work load of any calibrator is $\hat{c}$, i.e. $C_i[\omega]\leqslant\hat{c}$, $\forall(i,\omega)$.

The **total cost** of any iteration ω, denoted $C[\omega]$, is the weighted sum of: (a) C_{it}, the constant iteration overhead, (b) $C_w[\omega]$, the total movement time of all mobile calibrators, and (c) $C_c[\omega]$, the total calibration time at selected sensors:

$$C[\omega] = \mu_0\cdot C_{\text{it}} + \mu_w\cdot C_w[\omega] + \mu_c\cdot C_c[\omega] \tag{7}$$

where $C_w[\omega]{=}\sum_i C_{w,i}[\omega]$ and $C_c[\omega]{=}\sum_i C_{c,i}[\omega]$. Since we assume that each spot is only visited once by one calibrator, the total calibration cost is computed directly from $\mathbf{\Gamma}[\omega]$, i.e.:

$$C_c[\omega] = \sum_{l=1}^{L} \max_{j,k} \Big(D_{j,l}{\cdot}\Gamma_{j,k}[\omega]{\cdot}\tau_k \Big) \tag{8}$$

C. Multi-Sensor Calibration Planning Problem

We now introduce the **multi-sensor calibration planning** problem to minimize the **average cost** of operation over the maintenance period T. The problem is formulated as follows: Given the time span T, the map $\mathbf{G}$, the location matrix $\mathbf{D}$ and the sensor presence matrix $\mathbf{Q}$ of all the nodes, the calibration time τ_k and the calibration period T_k of all the sensor types, and the initial TTNC matrix $\mathbf{F}[1_-]$; find the total number of iterations Ω, and for each iteration $\omega{=}1,2,...,\Omega$, find the time t_ω it takes place, the sensor selection $\mathbf{\Gamma}[\omega]$, and the number and the paths of calibrations $\{\mathbf{W}[\omega]\}$; such that the **average cost** of all iterations over the time span T is minimized:

$$\min\ \frac{1}{T}\cdot\sum_{\omega=1}^{\Omega} C(\mathbf{\Gamma}[\omega],\{\mathbf{W}[\omega]\}) \tag{9}$$

$$\begin{aligned}
\text{s.t. }\ & t_1 = 0\\
& \Gamma_{j,k}[\omega]\in\{0,1\},\ \ \forall\omega,\forall j,\forall k\\
& W_{l_1,l_2,i}[\omega]\in\{0,1\},\ \ \forall\omega,\forall i,\forall(l_1,l_2)\\
& F_{j,k}[\omega_-]\geqslant 0,\ \ \forall\omega,\forall j,\forall k\\
& F_{j,k}[\omega_+] > 0,\ \ \forall\omega,\forall j,\forall k\\
& \Gamma_{j,k}[\omega]\leqslant Q_{j,k},\ \ \forall\omega,\forall j,\forall k\\
& \mathbf{F}[\omega_+] = \mathbf{\Gamma}[\omega]\circ\mathbf{T}_N + (1-\mathbf{\Gamma}[\omega])\circ\mathbf{F}[\omega_-],\ \ \forall\omega\\
& \mathbf{F}[(\omega{+}1)_-] = \mathbf{F}[\omega_+]{-}\min\mathbf{F}[\omega_+],\ \ \forall\omega\\
& t_{\omega+1} = t_\omega + \min\mathbf{F}[\omega_+],\ \ \forall\omega
\end{aligned}$$

$$t_\Omega + \min\mathbf{F}[\Omega_+]\geqslant T \tag{10}$$

$$h_l[\omega] = \bigvee_{j=1}^{N}\bigvee_{k=1}^{K}\Gamma_{j,k}[\omega]\cdot D_{j,l},\ \ \forall\omega,\forall l \tag{11}$$

$$\sum_{i=1}^{M}\sum_{l=1}^{L} W_{l,l_0,i}[\omega] = h_{l_0},\ \ l_0{=}2,3,\ldots,L,\forall\omega \tag{12}$$

$$C_i(\mathbf{\Gamma}[\omega],\mathbf{W}_i[\omega])\leqslant\hat{c},\ \ \forall\omega,\forall i \tag{13}$$

$\{\mathbf{W}[\omega]\}$ are valid path(s): constraints in IV-B apply.

where $C[\omega]$ is the total cost of iteration ω given by Equation (7), which depends on the sensor selection $\mathbf{\Gamma}$ and the calibrators' paths $\{\mathbf{W}[\omega]\}$, i.e. $C[\omega]{=}C(\mathbf{\Gamma}[\omega],\{\mathbf{W}[\omega]\})$; obviously, it also depends on problem inputs (i.e. $\mathbf{G}$, $\mathbf{D}$, $\mathbf{Q}$, etc.) which are hidden for cleaner expressions. The unnumbered constraints above are related to the definition of sensor selection and TTNC in Section III-A. Constraint (10) says the iterations need to cover the entire time span of T; (11) and (12) make sure all the spots with selected sensors in $\mathbf{\Gamma}$ are visited in $\{\mathbf{W}\}$; (13) says no calibrator should work for longer than $\hat{c}$ in any iteration. Additional constraints apply to ensure $\{\mathbf{W}\}$ are valid path(s).

Algorithm 1: TTNI-driven single-iteration local optimization algorithm for the sensor selection planning problem.

```
1  function solveSSPSingle ($\mathbf{G}$, $\mathbf{D}$, $\boldsymbol{\tau}$, $\mathbf{T}$, $\mathbf{Q}$, $\mathbf{F}$, $\boldsymbol{\mu}$, $\hat{c}$) ;
   Input  : $\boldsymbol{\tau}$ – $[\tau_k]$; $\mathbf{T}$ – $[T_k]$ $k{=}1,\ldots,K$;
            $\boldsymbol{\mu}$ – $\{\mu_0,\mu_w,\mu_c\}$;
            Refer to Section III-A, III-C for other symbols.
   Output: $\mathbf{\Gamma}$ – Sensor selection matrix.
2  tMin ← $\min\{F_{j,k} \mid Q_{j,k}{=}1\wedge F_{j,k}{>}0\}$ ; tMax ← $\min T_k$ ;
3  tCandSet ← $\{F_{j,k} \mid$ tMin$\leqslant F_{j,k}\leqslant$tMax$\}$ ;
4  Initialize minCostAvg ← $+\infty$ ; minGamma ← null ;
5  for each $T_{cand}$ in tCandSet do
6      $\mathbf{\Gamma} \leftarrow \mathbf{0}_{N\times K}$ ;
7      for j in $1,\ldots,N$ ; k in $1,\ldots,K$ do
8          $\Gamma_{j,k} \leftarrow (Q_{j,k}{=}1\wedge F_{j,k}{<}T_{\text{cand}}$ ;
9          for j' in $1,\ldots,N$ ; k' in $1,\ldots,K$ do
10             if $D_{j,l}{=}D_{j',l},\forall l$ then
11                 $\Gamma_{j',k'} \leftarrow (Q_{j',k'}{=}1\wedge\tau_{k'}\leqslant\tau_k)$ ;
12     $H \leftarrow \{l \mid \exists(j,k)$ s.t. $\Gamma_{j,k}{=}1\wedge D_{j,l}{=}1\}$ ; $\boldsymbol{\beta}\leftarrow\mathbf{0}_L$ ;
13     for l in $1,\ldots,L$ do $\beta_l \leftarrow \max_{j,k}(D_{j,l}{\cdot}\Gamma_{j,k}{\cdot}\tau_k)$ ;
14     cost ← $\mu_0{\cdot}C_{\text{it}} + \mu_c{\cdot}C_c(\mathbf{D},\mathbf{\Gamma},\boldsymbol{\tau})$ +
           $\mu_w{\cdot}C_w(\mathbf{G},$ solveMPPGreedy$(\mathbf{G},\hat{c},H,\boldsymbol{\beta}))$ ;
15     if (costAvg ← cost/$T_{cand}$) < minCostAvg then
16         minCostAvg ← costAvg ; minGamma ← $\mathbf{\Gamma}$ ;
17 return $\mathbf{\Gamma}$ ← minGamma ;
```

The sensor calibration planning problem is NP-hard. It tries to minimize the total cost of all iterations while the choices of early iterations can affect and limit the choices of later ones. Also, the cost of each iteration $C[\omega]$ involves a **movement time** $C_w[\omega]$, which also needs to be minimized, and thus requires an optimization on the paths of the calibrators, which is a variant of the Multiple Travelling Salesman Problem (mTSP) that is known to be NP-hard.

IV. Solutions and Algorithms

Our formulation in Equations (9–13) suggests we find $\mathbf{\Gamma}[\omega]$ (sensor selection) and $\{\mathbf{W}[\omega]\}$ (path plan) simultaneously for all iterations. However, we observe the fact that if we know which spots the calibrators need to visit, we can optimize the paths to visit them accordingly. Hence, instead of attempting to minimize $\sum C/T$, for each iteration ω, we attempt a two-phase local optimization on the **single-iteration average cost**, $C[\omega]/(t_{\omega+1}{-}t_\omega)$, where we decouple the optimization of $\mathbf{\Gamma}[\omega]$ and $\{\mathbf{W}[\omega]\}$. Accordingly, for each iteration we have a **sensor selection planning** phase and a **multi-path planning phase**. In the selection planning phase, given the initial TTNC matrix $\mathbf{F}[\omega_-]$, we optimize the sensor selection $\mathbf{\Gamma}$, from which we derive the set of selected spots $H{=}\{h_l \mid h_l{=}1,\forall l\}$, which is then used in the path planning phase to decide the number of calibrators and the optimal path(s) to visit the selected spots.

A. Sensor Selection Planning Algorithms

Leveraging the discrete nature of TTNC and the definition of TTNI (time to next iteration), we propose the TTNI-driven local optimization algorithm. The intuition behind this algorithm is to exhaust the possible values of TTNI (i.e. $t_{\omega+1}-t_{\omega}$) and find the "cheapest" one to fulfill.

The procedure of the TTNI-driven local optimization is shown in Algorithm 1. It involves the following steps: (1) Determine **all the possible values of TTNI** that could result from any possible sensor selection in this iteration. The minimum TTNI candidate is $\min\{F_{j,k}[\omega_-] \mid Q_{j,k}{=}1 \wedge F_{j,k}[\omega_-]{>}0\}$, selecting only the sensors that need immediate calibration (Ln 2). The maximum TTNI candidate is $\min T_k$, selecting **all sensors** (Ln 2). All values in $\mathbf{F}[\omega_-]$ between them become TTNI candidates (Ln 3). In the worst case, the number of TTNI candidates is $O(N{\cdot}K)$ (2) For each TTNI candidate T_{cand}, tentatively assume it to be the desired TTNI and create the minimum selection of sensors to meet the TTNI, i.e. let $\Gamma_{j,k}{=}1$ if $Q_{j,k}{=}1$ and $F_{j,k}[\omega_-]{<}T_{\text{cand}}$ (Ln 5–8); then add all the sensors that are co-located with the selected sensors and that do not induce extra time for calibration (Ln 9–11, because their calibration is done in parallel, if it takes a shorter time). Generating $\mathbf{\Gamma}$ from T_{cand} takes $O(N{\cdot}K{+}N{\cdot}L)$ time. Compute the single-iteration average cost from $\mathbf{\Gamma}$ (Ln 12–14). (3) Select the TTNI candidate that gives the minimum average cost (Ln 16), and its corresponding $\mathbf{\Gamma}$ is the output of the algorithm. The worst-case running time excluding the time used to compute or estimate the movement time, is $O(N^2{\cdot}K^2{+}N^2{\cdot}K{\cdot}L)$.

If during step (2) we are able to compute the optimal paths of calibrators, we will compute the best cost evaluation for each selection and find the local optima. Unfortunately, multi-path planning is also NP-hard. We then propose two heuristics: a fast nearest-neighbor-based greedy algorithm, and an improved genetic algorithm (GA). During sensor selection, we use the faster greedy algorithm to estimate the movement cost; once the selection is done, we use GA to generate the final path(s) for the iteration.

B. Multiple-Path Planning Algorithms

The **multi-path planning problem** for a specific iteration ω refers to a sub-problem in our two-phase local optimization solution to the sensor calibration planning problem. The objective is to generate a set of paths $\{\mathbf{W}[\omega]\}$ of minimum cost (i.e. movement time $C_w[\omega]$) for the selected spots yielded by the sensor selection $\mathbf{\Gamma}[\omega]$. It is a variant of the classic mTSP or VRP: we determine the number of calibrators based on the demand instead of having the number m of travellers given, as in mTSP. Also, evaluating the calibrator workload constraint involves the movement time of individual calibrators, which adds to the complexity of solutions.

Hence, we derive the following mixed-integer-programming (MIP) formulation of the multi-path planning problem based on a flow-based three-index MIP formulation of mTSP [25], adding appropriate modifications to match our assumptions and constraints: Given a map $\mathbf{G}$, the location of the nodes $\mathbf{D}$, the sensor selection $\mathbf{\Gamma}$, and the calibration time τ_k, $\forall k$; find $\mathbf{W}_{L\times L\times M}$ (represented in a more general form that could contain calibrators with no assignment) and helper variables $\mathbf{U}_{L\times M}$ to

$$\min \sum_{l_1=1}^{L}\sum_{l_2=1}^{L}\left(G_{l_1,l_2}\cdot\sum_{i=1}^{M}W_{l_1,l_2,i}\right) \tag{14}$$

$$\begin{aligned}
\text{s.t. } & W_{l_1,l_2,i}\in\{0,1\},\ \forall i,\forall(l_1,l_2)\\
& W_{l,l,i}=0,\ \forall l{=}2,3,\dots,L,\forall i\\
& \textstyle\sum_{l_2=1}^{L}W_{1,l_2,i}=1,\ \forall i\\
& \textstyle\sum_{l_1=1}^{L}W_{l_1,l,i}-\sum_{l_2=1}^{L}W_{l,l_2,i}=0,\ \forall i,\forall l\\
& \textstyle\sum_{l_1=1}^{L}\sum_{i=1}^{L}W_{l_1,l_2,i}=h_l,\ \forall l_2 && (15)\\
& u_{l,i}\geqslant 2,\ \forall i,\forall l\\
& u_{l_1,i}-u_{l_1,i}+1-(L-1)\cdot(1-W_{l_1,l_2,i})\leqslant 0,\\
& \qquad\qquad\qquad\qquad \forall i,\forall(l_1,l_2)\\
& \textstyle\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}W_{l_1,l_2,i}\cdot\big(G_{l_1,l_2}+\Upsilon_{l_2}\big)\leqslant\hat{c},\ \forall i && (16)
\end{aligned}$$

Where $W_{l_1,l_2,i}{=}1$ if calibrator m_i visits spot ν_{l_2} immediately after spot ν_{l_1}; or 0 otherwise. M is the maximum number of calibrators; assuming we always have enough calibrators, L would be an effective upper bound of M to be used in solvers. The unnumbered constraints are related to the construction of multiple paths. Constraint (15) makes sure all selected spots are visited by exactly one calibrator; (16) enforces the maximum workload of calibrators, where Υ_l is the total calibration time spent at spot ν_l, i.e. $\Upsilon_l[\omega]=\max_{j,k}(D_{j,l}{\cdot}\Gamma_{j,k}[\omega]{\cdot}\tau_k)$.

This formulation allows us to apply MIP solvers directly. However, the problem is NP-hard; the number of independent variables and the number of constraints in this MIP formulation are both in the order of $O(L^3)$, resulting in a huge solution space. It is hard for any MIP solver to optimally solve the problem in a reasonable amount of time [25]. In particular, we tried two widely used solvers: GLPK (GNU Linear Programming Kit, open-source – https://www.gnu.org/software/glpk/) and Gurobi (commercial software – http://www.gurobi.com/). None of the two solved the problem in less than 48 hours for $L{\geqslant}15$. To solve the problem at larger scale, we propose two heuristics: a greedy algorithm derived from the nearest neighbor heuristic of traditional TSP, and an improved genetic algorithm (GA) based on the one proposed by Sedighpour, et al. [26] for mTSP. For a clean design of the algorithms, after the completion of the sensor selection planning phase, the planning framework computes the set of **selected spots** H and the calibration time $\boldsymbol{\beta}$ at these spots from the sensor selection matrix $\mathbf{\Gamma}$.

1) Nearest-Neighbor-Based Greedy Heuristic: The nearest neighbor algorithm for TSP starts with a tour containing only one spot. At each step, it determines that the next spot to visit as the one that is closest to the last visited spot, and loops until all the spots are visited.

Inspired by this straightforward TSP algorithm, we derive our greedy algorithm for the multi-path planning problem shown in Algorithm 2 as follows: (1) Start with a set of empty paths (i.e. all the calibrators stay at the depot) and the set of all selected spots H. (2) At each step, for every unvisited spot, compute the extra travel and calibration time yield by adding it to the end of the path of each mobile calibrator as long as

Algorithm 2: Nearest-neighbor-based greedy algorithm for the multi-path planning problem.

```
1  function solveMPPGreedy ($\mathbf{G}$, $\hat{c}$, $H$, $\boldsymbol{\beta}$) ;
   Input  : $\mathbf{G}$ – Map; $\hat{c}$ – Maximum workload;
            $H$ – Set of selected spots;
            $\boldsymbol{\beta}$ – Vector of calibration time at selected spots.
   Output: $\{\mathbf{W}\}$ – Set of paths.
2  Initialize pathSet $\leftarrow$ [ ] ;
3  while $H$ is not empty do
4      minInc $\leftarrow +\infty$ ; minSp $\leftarrow$ minPath $\leftarrow$ null;
5      for each path in pathSet ; last $\leftarrow$ path$[-1]$ do
6          oldTime $\leftarrow$ getMoveTime (path) + $\sum_{l\in \text{path}} \beta_l$ ;
7          for each sp in $H$ do
8              dCw $\leftarrow G[\text{last}, \text{sp}] + G[\text{sp}, 1] - G[\text{last}, 1]$ ;
9              if oldTime + dCw + $\beta_{sp} \leqslant \hat{c}$ then
10                 if dCw < minInc then
11                     minInc $\leftarrow$ dCw ;
12                     minSp $\leftarrow$ sp ; minPath $\leftarrow$ path ;
13     if minInc is finite then minPath.append (minSp) ;
14     else
15         for each sp in $H$ do
16             if (dCw $\leftarrow G[1, sp] + G[sp, 1]$) < minInc then
17                 minInc $\leftarrow$ dCw ; minSp $\leftarrow$ sp ;
18         newPath $\leftarrow$ [minSp] ; pathSet.add (newPath) ;
19     $H$.del (minSp) ;
20 return $\{\mathbf{W}\} \leftarrow$ convertPathVecToMatrix (pathSet) ;
```

the calibrator is not overloaded. Pick the spot-calibrator pair that induces the least additional movement time. Note that the spot could be added to an old calibrator (Ln 5–13) or a new calibrator (Ln 15–18). (3) Loop until all the spots are visited (Ln 3, 19). Note: Our actual implementation of this algorithm caches the movement and calibration time associated with each calibrator to reduce redundant computation, so the worst-case running time of this algorithm is $O(L^3)$.

2) Improved Genetic Algorithm (GA): We design our genetic algorithm (GA) based on the mTSP GA solution of [26]. Features are added to address the peculiarities of our MPP formulation, i.e. the variable number of calibrators, the workload constraint, and the map (i.e. a directed-graph).

A **chromosome** is an integer vector that is made of two parts: a permutation of all the selected spots (1st half) and an assignment mapping the spots to mobile calibrators (2nd half). If the number of selected spots is $|H|{=}L'$, a chromosome will have length $2L'$. The assignment (2nd half) is represented by the number of spots visited by each calibrator, so these integers should all be in range $[0, L']$ and sum up to L'. For example, chromosome $[2, 4, 5, 6, 3, 2, 3, 0, 0, 0]$ means $L'{=}5$ and that m_1 will visit spots ν_2, ν_4, and m_2 will visit ν_5, ν_6, ν_3. The **fitness** is the negative of the total movement time of all the mobile calibrators, and the **selection** is done by a standard scaled-fitness proportional selection.

The **initial population** is composed of randomly generated individuals. The permutation is performed by a uniformly random permutation generator, and the assignment is done by uniformly and randomly picking an integer and subtracting it from the total number of selected spots until none is left. The **crossover** is done by applying a standard "order crossover" on the first half of the chromosome.

Because of the variable number of mobile calibrators, we design three helper functions that apply to chromosomes: (a) **compress**: shift all zeros in the assignment section to the end and non-zeros values to the beginning; (b) **split**: check if any assignment ($\geqslant$2 spots) leads to an overloaded calibrator, randomly split it into two calibrators, and loop until none is found; (c) **merge**: check if there exists a pair of assignments that can be merged into one without overloading the calibrator; then merge the first pair found. Among the three, **compress** and **split** are applied to every newly-generated chromosome during population initialization, mutation, and crossover, while **merge** is applied as one type of mutation.

Apart from "merge", there are three other types of **mutation**: (a) two-point swap, (b) segment reversal, and (c) 3-opt local optimization. Every time a mutation is triggered, we randomly pick one of the four types of mutation functions. (a) and (b) are straightforward. 3-opt [27] is a local optimization for TSP, which tries to break a tour into three segments by removing three edges, and reconnect the three segments into a new but shorter tour. 2-opt is a commonly used local optimization for TSP on undirected graphs, but an odd numbered opt is required for digraphs to avoid reversing any segment, which makes it faster to compute the new movement time.

Finally, the tunable parameters such as the population size, elite-keeping size, and the termination conditions are assigned by the framework according to the problem size (i.e. L').

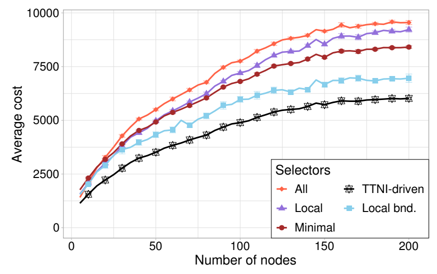
V. Validation

We evaluate the performance of our proposed multi-sensor calibration planner using realistic data derived from testbeds and present the steps we are taking towards a usable system

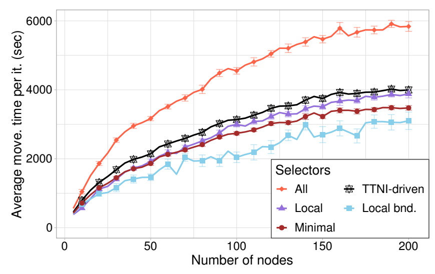
TABLE I
EXPERIMENTAL SETUP FOR THE PERFORMANCE EVALUATION.

Input Data			Indoor		Outdoor
			Normal	**Emergency**	
Map	L	Num. of Spots	60		63
	G	Pairwise Dist.[a]	$\leqslant$53 sec		$\leqslant$73 min
Sensor	K	Num. of Types	10		8
	τ	Calib. Time	1–30 min		0.25–1 min
	T	Calib. Period	14–91 d	7–91 d	28–123 d
Nodes	N	Num. of Nodes	100 (varies if independent variable)		
		Sensor Presence	50 (varies if independent variable)		
User Req.	$\hat{c}$	Max. Workload	2 hours		4 hours
	$\boldsymbol{\mu}$	Coefficients	C_{it}=10000, μ_0=1 μ_w=5, μ_c=1		
	T	Maintenance P.	360 days		

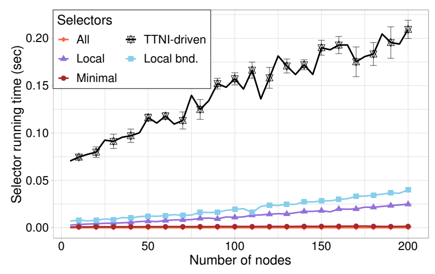
[a] Pairwise distance correspond to the shortest traveling time.



(a) Average cost vs. number of nodes.

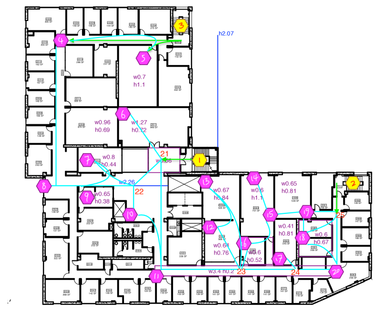


(b) Avg. move. time (s) per iter. vs. numb. of nodes.



(c) Mean running time (sec) vs. number of nodes.

Fig. 3. Impact of the number of nodes on the sensor selection algorithms in an indoor setup, with sensor presence rate $\sum \mathbf{Q}/(N{\cdot}K)$=0.5.


(a) DBH 2nd floor; data set contains all six floors; real and synthetic spots.

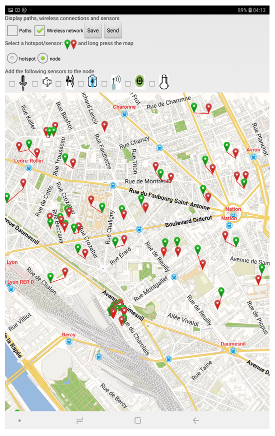

(b) Paris area of 10 km^2; synthetic spots.

Fig. 4. Smart space structure and spot location used in evaluation.

for calibration planning This includes modules to facilitate the modeling of the (indoor/outdoor) environment and to provide navigation guidance to the calibrators (via an Android app).

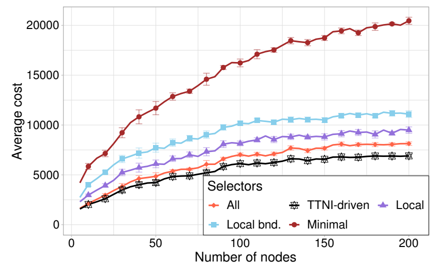
Experimental Setup: We conduct a series of evaluations using three sets of input data. The two first involve the instrumented building at UC Irvine (Figure 4a) we discussed in Section II, which is used for everyday monitoring (normal condition) and for supporting emergency operations when needed. The desired calibration frequency is a parameter that is learned from empirical study and that depends on the context. We consider the actual deployment in our real testbeds and generate additional spots (with sensors and nodes) using a similar pattern. Using the service we present in Section V, we synthesized the third data set that relates to an outdoor urban environment in which sensors are placed to monitor noise and air quality. The parameters are summarized in Table I.

We compare our sensor selection with two naïve sensor selection strategies that aim at "always selecting *all* sensors" (regardless of their TTNC) and "only selecting the *minimal* set of sensors" (i.e. those we must calibrate because their TTNC reaches 0). In addition, we also investigate two simple selection strategies. The former consists in selecting all sensors that are co-located with the sensors that form the minimal set ("*local*"). The latter consists in only selecting the sensors that can automatically calibrate with each other without human intervention and that henceforth do not induce additional calibration time ("*local bounded*"). For multi-path planning, we evaluate the performance of two MIP solvers (*GLPK* and *Gurobi*), our two path planning heuristics (NN-based *greedy* and *GA*), and a naïve strategy that sends one calibrator to *each* spot and that should give the highest cost. Algorithm running time is evaluated on the OpenLab cluster of Dept. Computer Science at UCI, where each computing node has 2x Quad-core Intel Xeon 3.0GHz CPU E5450 CPUs.

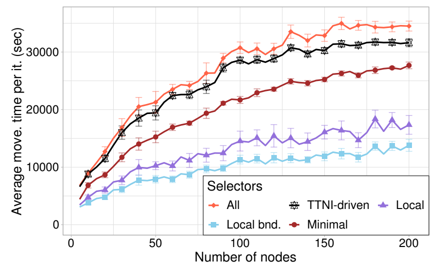
Indoor vs. Outdoor Results: In an indoor environment (Figure 3), our algorithm (TTNI-driven sensor selection and GA-based multi-path planning) always result in a lower average cost for N ranging from 5 to 200. Compared to the naïve sensor selection strategies, such as "selecting **all** sensors" and "selecting the **minimal** set of sensors" (still considering GA-based multi-path planning), our algorithm combination provides up-to 30% improvement in the long-term average cost. Note that even though our algorithm does not always end up with the lowest cost per iteration (Figure 3b), it makes a fair trade-off between the cost and the time (between iterations). Figure 3c shows that the time spent to select sensors is short – less than 1 sec for a reasonably complex building incorporating 200 nodes and 60 spots. The same trend also applies in the outdoor environment (Figure 5), where the distances between spots are significantly longer (Table I). As the spatial span of the setup grows, the difference among the algorithms becomes more dramatic (note the different y-scale in Figures 3a,3b and 5a,5b). Certain naïve approaches are very sensitive to this change (Figure 5b, "minimal" and "local bounded"), while our algorithm shows stable performance in both settings.

Normal vs. Emergency Condition Results: Having demonstrated the effectiveness of our algorithm in indoor and outdoor settings, we further study the performance of our approach in an emergency scenario where the calibration requirements of certain sensor types are increased (Figures 6a and 6c, note the difference in y-scale). When calibration is required more frequently for some sensors, the naïve/simple approaches suffer from a big increase in average cost, especially when the sensors are deployed densely ($\sum \mathbf{Q}/(N{\cdot}K)$>0.5), while the performance of our algorithm and "local bounded" are less affected. We also notice that unlike the simpler approaches ("local bounded" or "minimal"), our TTNI-driven sensor selection algorithm avoids the desynchronization of the periodical calibrations, while the "local bounded" strategy does so with a small number of sensors (Figure 6b).

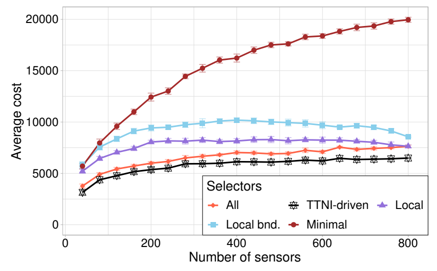
Scalability Results for the Multi-Path Planning: Figure 7



(a) Average cost vs. numb. of sensors N; sensor presence $\sum \mathbf{Q}/(N{\cdot}K)$=0.5; outdoor scenario.

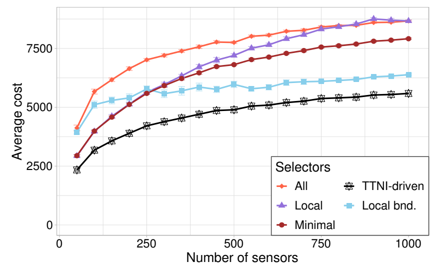


(b) Average movement time (sec) per iteration vs. N; $\sum \mathbf{Q}/(N{\cdot}K)$=0.5; outdoor scenario.

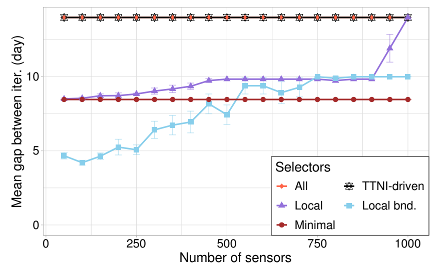


(c) Average cost vs. number of sensors $\sum \mathbf{Q}$; N=100; outdoor scenario.

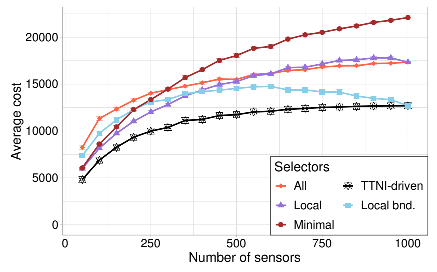
Fig. 5. Evaluation of the sensor selection algorithms in an outdoor scenario.



(a) Average cost vs. number of sensors $\sum \mathbf{Q}$; normal scenario.

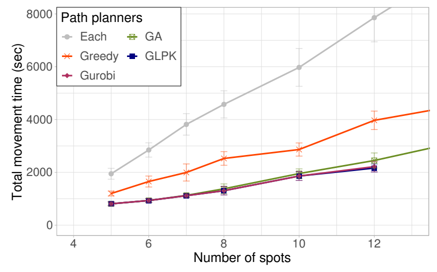


(b) Mean interval (day) between iterations vs. number of sensors; normal condition.

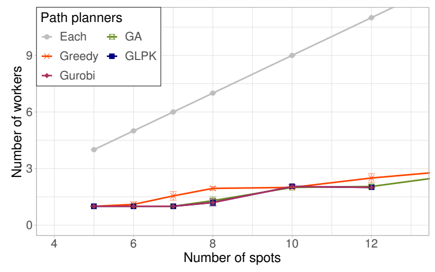


(c) Average cost vs. number of sensors; emergency condition.

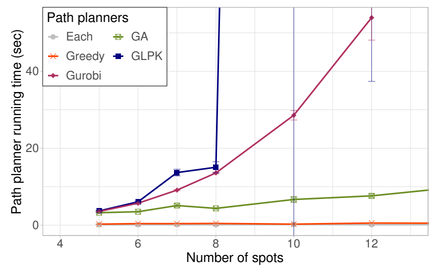
Fig. 6. Impact of the number of sensors on the sensor selection in an indoor environment containing 100 nodes.



(a) Movement time vs. number of spots.



(b) Number of calibrators vs. number of spots.

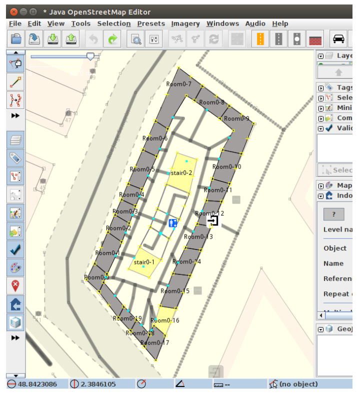


(c) Running time (sec) vs. number of spots

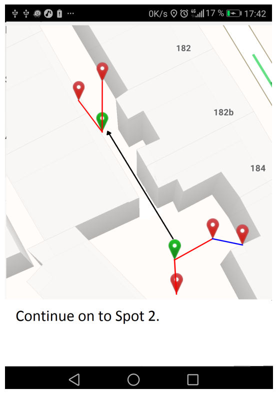
Fig. 7. Scalability of the multi-path planning solvers and proposed heuristic algorithms.

compares the performance of multi-path planning algorithms for the number of spots $L\leqslant 12$: GA provides close-to-optimal solutions but takes 8–10 sec for L=12 (20–60 sec for L=60); the greedy heuristic is much faster (approx. 0.5 sec for L=60), which makes it suitable as an estimator during the sensor selection planning. GLPK and Gurobi (i.e. the MIP solvers) could not terminate within 48 hours for $L\geqslant 15$ so we aborted.

Towards a Usable System: Creating a calibration planning service is not trivial. In addition to the algorithm for planning, we are implementing a service to enable flexible calibration. To support usability, we have implemented a toolkit that serves to generate 3-dimensional maps and provides navigation guidance to the mobile calibrators. We create the 3-D navigable maps using OpenStreetMap (www.openstreetmap.org), which provides a basemap of the outdoor environment, e.g., roads (lines), junctions (points), buildings (polygons). The basemap is further edited (Figure 8a) so as to (i) detail the inner structure of the buildings of interest (if any), and (ii) layer a number of features including the paths accessible in the indoor environment. To design the inner structure of the building, we rely on JOSM (josm.openstreetmap.de). Following, we extract


Java OpenStreetMap Editor
File Edit View Tools Selection Presets Imagery Windows Audio Help
Room0-7
Room0-6
Room0-5
stair0-2
stair0-1
Room0-16
Room0-17
48.8423086
2.3846105
(no object)

(a)



Continue on to Spot 2.

(b)

Fig. 8. (a) Modeling indoor space in a building and (b) Mapping of a indoor/outdoor space: Spots (green markers) are placed; nodes (red markers) compose a spot and communicate through wireless links (blue lines).

a representation of the connection graph from the resulting map using GraphHoper (www.graphhopper.com). This graph is composed of edges corresponding to streets, roads as well as indoor-pedestrian paths, and nodes representing junctions.

The next step consists in placing on the map, the spots that should be visited along with the nodes and the related sensors. The wireless connections among sensing nodes are also modeled to inform the propagation of calibration parameters. The map are exploited by the Android app that we implemented for assisting the mobile workers. The app computes the shortest path (avoiding obstacles) between any 2 spots and provides this information to the multi-sensor calibration planner. For instance, with 60 spots and 77 nodes equipped by 188 sensors mapped over an urban area of 10 km^2, the resulting connection graph is generated in 2.967s. This graph, which contains 8768 edges (e.g., streets) and 5661 vertices (e.g., junctions), is further used to compute the paths between any two spots; such a computation takes in average 0.0195s for an average distance separating two spots of 1387 meters. Relying on the multi-sensor calibration planner, the mobile app further provides navigation guidance to the calibrators (Figure 8b).

VI. Conclusion

Effective deployment of IoT remains a challenging task with a multitude of pitfalls: once deployed, sensors are subject to drifts, bias and thereby fail to provide the expected and meaningful data. Cost-effectively planning the on-site calibration of IoT devices helps in the sustainable long term operation of deployments. Our work focuses on collaborative and distributed maintenance of an IoT-based system. We build on our experiences in deploying sensors over smart spaces and on addressing the calibration tasks to enhance the quality of the gathered observations. We frame multi-sensor calibration planning as an optimization problem where we propose a two-phase iterative local optimization approach to determine (a) how many calibration iterations are necessary, (b) which sensors should be calibrated at each of these iterations, and (c) the number of mobile calibrators that are required (as well as their respective calibration paths), such that the average cost of all iterations is minimized and under the constraint that the calibrators should not be overloaded. Our evaluation shows that the proposed algorithms solve the calibration planning problem effectively compared to naïve/simple solutions.

Future Work: Introducing a service to facilitate large-scale calibration in IoT deployments is promising; while our initial studies demonstrate the value and feasibility of this approach, long-term studies with an end-to-end system will help adapt and fine-tune the calibration process for different applications and external dynamicity.

Acknowledgement: The authors thank all members of the ISG and DSM groups at UCI for the valuable discussions. We also thank Inria for supporting the MINES associate team. This project is funded under NSF award CNS-1450768 and NIST award 70NANB17H285.

References

[1] G. Kortuem, F. Kawsar *et al.*, "Smart objects as building blocks for the internet of things," *IEEE Internet Computing*, vol. 14, no. 1, 2010.

[2] D. Uckelmann, M. Harrison *et al.*, *An Architectural Approach Towards the Future Internet of Things*. Springer, 2011.

[3] O. Vermesan and P. Friess, Eds., *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*, ser. Publishers Series in Communication. River, 2013.

[4] L. Atzori, A. Iera *et al.*, "The internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, Oct. 2010.

[5] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, 2014.

[6] D. Stone, "Calibration and linear regression analysis: A self-guided tutorial."

[7] S. Nesamony, M. K. Vairamuthu *et al.*, "On optimal route of a calibrating mobile sink in a wireless sensor network," in *IEEE INSS*, 2007.

[8] S. Hachem, V. Mallet *et al.*, "Monitoring Noise Pollution Using The Urban Civics Middleware," in *IEEE BigDataService*, 2015.

[9] V. Issarny, V. Mallet *et al.*, "Dos and Don'ts in Mobile Phone Sensing Middleware: Learning from a Large-Scale Experiment," in *ACM/IFIP/USENIX Middleware*, Dec. 2016.

[10] R. Ventura, V. Mallet *et al.*, "Evaluation and calibration of mobile phones for noise monitoring application," *Journal of the Acoustical Society of America*, vol. 142, no. 5, 2017.

[11] ——, "Assimilation of mobile phone measurements for noise mapping of a neighborhood," *Journal of the Acoustical Society of America*, vol. 144, no. 3, 2018.

[12] F. Sailhan, V. Issarny *et al.*, "Opportunistic multiparty calibration for robust participatory sensing," in *IEEE MASS*, 2017.

[13] K. Benson, C. Fracchia *et al.*, "SCALE: Safe community awareness and alerting leveraging the internet of things," *IEEE Communications Magazine*, vol. 53, no. 12, 2015.

[14] M. Y. S. Uddin, A. Nelson *et al.*, "The scale2 multi-network architecture for iot-based resilient communities," in *IEEE SMARTCOMP*, 2016.

[15] M. Rahman, H.-J. Hong *et al.*, "Adaptive sensing using internet-of-things with constrained communications," in *Proceedings of the 16th Workshop on Adaptive and Reflective Middleware*. ACM, 2017.

[16] M. Rahman, A. Rahman *et al.*, "An adaptive IoT platform on budgeted 3g data plans," *Journal of Systems Architecture*, 2018.

[17] S. Mehrotra, A. Kobsa *et al.*, "Tippers: A privacy cognizant iot environment," in *IEEE PerCom Workshops*. IEEE, 2016.

[18] G. Barrenetxea, F. Ingelrest *et al.*, "The hitchhiker's guide to successful wireless sensor network deployment," in *ACM SenSys*, 2008.

[19] K. Ni, N. Ramanathan *et al.*, "Sensor network data fault types," *ACM Transactions on Sensor Networks*, vol. 5, no. 3, 2009.

[20] M. S. Stankovic, S. S. Stankovic *et al.*, "Asynchronous distributed blind calibration of sensor networks under noisy measurements," *IEEE Transactions on Control of Network Systems*, 2016.

[21] Y. Wang, A. Yang *et al.*, "A deep learning approach for blind drift calibration of sensor networks," *IEEE Sensors Journal*, 2017.

[22] D. Hasenfratz, O. Saukh *et al.*, "On-the-fly calibration of low-cost gas sensors," *Wireless Sensor Networks*, 2012.

[23] O. Saukh, D. Hasenfratz *et al.*, "Reducing multi-hop calibration errors in large-scale mobile sensor networks," in *ACM IPSN*, 2015.

[24] K. Fu, W. Ren *et al.*, "Multihop calibration for mobile sensing: K-hop calibratability and reference sensor deployment," in *INFOCOM*, 2017.

[25] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, 2006.

[26] M. Sedighpour, M. Yousefikhoshbakht *et al.*, "An effective genetic algorithm for solving the multiple traveling salesman problem," *Journal of Optimization in Industrial Engineering*, no. 8, 2012.

[27] K. D. Boese, *Cost versus distance in the traveling salesman problem*. UCLA Computer Science Department Los Angeles, 1995.

[28] J. Thelen, D. Goense *et al.*, "Radio wave propagation in a patatoe field," in *1st workshop on wireless network measurement*, 2005.

[29] R. Tan, G. Xing *et al.*, "Adaptive calibration for fusion-based wireless sensor networks," in *INFOCOM*, 2010.

[30] F. Sailhan, V. Issarny *et al.*, "Opportunistic multiparty calibration for robust participatory sensing," in *IEEE MASS*, 2017.

[31] R. Vaisenberg, S. Ji *et al.*, "Exploiting semantics for sensor re-calibration in event detection systems," in *Proc. of SPIE*, vol. 6818, 2008.

[32] E. Miluzzo, N. D. Lane *et al.*, "Calibree: A self-calibration system for mobile sensor networks," in *DCOSS*. Springer, 2008.

[33] L. Balzano and R. Nowak, "Blind calibration of sensor networks," in *ACM IPSN*, 2007.

[34] N. Ramanathan, L. Balzano *et al.*, "Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks," *Center for Embedded Network Sensing*, 2006.

[35] J. Gupta, *et al.*, "Comparison of some approximate algorithms proposed for traveling salesmen and graph partitioning problems," in *ICIoTCT*, 2018.