Spatiotemporal Scheduling for Crowd Augmented Urban Sensing

Qiuxi Zhu, Md Yusuf Sarwar Uddin, Nalini Venkatasubramanian
University of California, Irvine
Email: qiuxiz@ics.uci.edu, yusuf.sarwar@uci.edu, nalini@ics.uci.edu

Cheng-Hsin Hsu
National Tsing Hua University
Email: chsu@cs.nthu.edu.tw

***Abstract*—In urban environments, mobile crowdsensing can be used to augment in-situ sensing deployments (e.g. for environmental and community monitoring) in a flexible and cost-efficient manner. The additional participation provided by crowdsensing enables improved data collection coverage and enhances timeliness of data delivery. However, as the number of participating devices/users increases, efficient management is required to handle the increased operational cost of the infrastructure and associated cloud services – exploiting spatiotemporal redundancy in sensing can help cost-efficient utilization of resources. In this paper, we develop solutions to exploit the mobility of the crowd and manage the sensing capability of participating devices to effectively meet application/user demands for hybrid urban sensing applications. Specifically, we address the spatiotemporal scheduling problem to create high-resolution maps (e.g. for pollution sensing) by developing a common framework to capture spatiotemporal impact of multiple sensor types that generate heterogeneous data at different levels of granularity. We develop an online scheduling approach that leverages the knowledge of device location and sensing capability to selectively activate nodes and sensors. We build a multi-sensor platform that enables data collection, data exchange, and node management. Prototype deployments in three different campus/community testbeds were instrumented for measurements. Traces collected from the testbeds are used to drive extensive large scale simulations. Results show that our proposed solution achieves improved data coverage and utility under data constraints with lower costs (30% fewer active nodes) than naive approaches.**

I. Introduction

A new wave of information systems has been enabled by the gigantic improvements in the capabilities of embedded systems and mobile computing platforms, developments in low-energy sensing technologies, the expanding coverage of wireless communication infrastructures, and the rapid growth of cloud-based solutions and services. The rising number of connected personal devices have led to a tighter engagement of users and communities through Internet-of-Things deployments and participatory crowd-sensing applications - this is indeed one of the key goals of smart city efforts around the world. Applications such as urban pollution monitoring [1]–[6] are beginning to leverage these technology trends.

While instrumenting community infrastructures (e.g. street lamps, traffic lights, buildings, roads) with IoT sensing/actuation capabilities provides awareness of current conditions, deployment limitations of in-situ devices lead to the lack of ubiquitous coverage. Our goal is to augment in-situ deployments in communities by leveraging crowd-sourced sensor information from mobile users who are "on-the-go" in urban areas. Several challenges arise in enabling mobile crowdsensing as flexible and cost-efficient extensions to existing stationary deployments - scalability and heterogeneity are two key issues. As deployments scale in the number of users and devices, there is increasing redundancy in data traffic, consequently increasing operational cost and resource constraints. Coordination of sensing tasks across multiple devices with diverse sensing capabilities and availabilities is required to manage the hybrid sensing [7] in an efficient manner.

One promising solution for the management of a hybrid (i.e. in-situ and mobile) environment is to leverage global knowledge (e.g. location and sensing capability of all participating devices) to selectively assign sensing tasks to participants so as to reduce the level of redundancy, while maintaining relatively high accuracy and spatiotemporal resolution of the collected data. Joint scheduling in such a hybrid configuration requires uniform concepts that capture (a) the diverse spatiotemporal needs of sensing applications and their associated costs, (b) the heterogeneity of devices (sensor types with varying spatial accuracy and compute capabilities), and (c) sensing phenomena that vary in their spatiotemporal extent and dictate the urgency of communication to target recipients.

In this paper, we address a novel spatiotemporal scheduling problem for crowd augmented urban sensing that supports the monitoring of multiple events in a community. We unify concepts across both in-situ and mobile sensors by defining the notion of *spatiotemporal impact* of sensor readings. We determine how to effectively assign sensing workloads to each participant in order to reduce data redundancy using spatial and temporal knowledge. The key idea is to leverage the knowledge of (a) application requirements (e.g. air pollution), which can be defined by application maintainers and community users; (b) device heterogeneity and mobility (collected at runtime); and (c) spatiotemporal properties of target variables (e.g. concentration of air pollutant), which can be obtained from theoretical models and in-field measurements. Using this knowledge, we propose an online scheduling technique that periodically generates globally optimized plans for all devices.

Application Scenario and Problem Description: To lend focus to the problem development and prototype environment design, we employ a targeted use case – *community-scale pollution monitoring*. Pollution monitoring has been a hot topic for decades, and it is increasingly important in densely

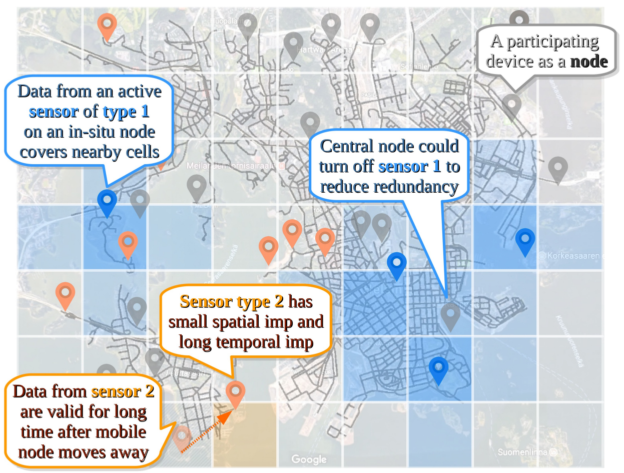

A participating device as a node
Data from an active sensor of type 1 on an in-situ node covers nearby cells
Central node could turn off sensor 1 to reduce redundancy
Sensor type 2 has small spatial imp and long temporal imp
Data from sensor 2 are valid for long time after mobile node moves away

Fig. 1. An urban crowdsensing scenario where we selectively activate sensors on participating nodes to reduce redundancy while providing good coverage.

populated urban areas. Studies [8]–[11] have shown significant correlation between urban air pollution and mortality, raising public awareness and concern [12]. Traditionally, urban air pollution has been captured by city-owned air quality monitoring stations. While they help us to collect data with higher accuracy, they are expensive to setup, operate, and maintain. Given the limited number of such stations [1], information is often at coarse levels of granularity. Today, with community IoT deployments and crowd-owned mobile devices, pollution data can be gathered at much higher spatiotemporal resolutions to support analytical and decision making applications. In-situ IoT deployments are valuable with better connectivity, power supply, and sometimes accuracy, while mobile devices augment the IoT deployment with extended coverage and additional sensing capability. We strive to create a uniform framework that leverages the characteristics of both modalities.

Our aim (i.e. applications) is to create *high-resolution maps* for multiple commonplace pollution modes (e.g. air, noise, trash), as well as other dynamically evolving phenomena (e.g. traffic, Wi-Fi) in urban environment, especially in a community-wide setting. The pollution monitoring system setup is as follows: Sensor nodes are equipped with varying types of sensors for pollution mapping. Each pollution type (e.g. air, noise, trash) can be characterized using spatiotemporal resolution requirements; sensors for these pollution types have a spatiotemporal impact (i.e. range and duration of sensor data validity) (Fig. 1). The task at hand is to determine how to control data collection to meet the requirements. A regional edge server is deployed for centralized control of devices and simple analyses on raw data. Due to practical bounds on the resources available at the edge server and sensors (e.g. CPU, bandwidth), our plan is to activate a subset of the sensors at any time while retaining the quality of the maps we create, i.e. the sensing activity of each node must be carefully scheduled.

More specifically, given information on the location and sensing capability of all participating nodes at any time, and the spatiotemporal characteristics of all pollution types, our goal is to determine which sensors on which nodes should be activated during a given time period, so that (a) the generated pollution map maximizes the space-time coverage; (b) the total amount of data received by the server is bound; and (c) the total number of workers stays low, so the system suffers from less uncertainty and energy overhead. This spatiotemporal scheduling problem is challenging for the following reasons: (a) Crowd participants move at will; we cannot control their movement or create plans a priori; (b) The varying resolution requirements from applications and the heterogeneous nature of crowd-owned devices make it hard to decouple them during scheduling. In this paper, we formalize this spatiotemporal scheduling as a constrained multi-objective optimization problem with discrete space-time representation.

Key Contributions of This Paper: Computing an optimal spatiotemporal plan for activating sensors and devices requires knowledge of all nodal states – this is infeasible given unpredictability of node movements in the future. Hence, we propose an online planning approach, where a broker (cloud resource or logically centralized edge server in a region) collects information from nodes periodically and generates an activation schedule for the near future. Key contributions of this paper include: (a) Formalization of spatiotemporal scheduling as a constrained multi-objective optimization problem (Section II), which is NP-hard. (b) Design of two online scheduling algorithms (Section III) that compute sensor activation plans iteratively using states of all nodes and corresponding historical data. (c) Development of prototype pollution sensing platforms (in-situ and mobile)with measurement studies in three real community testbeds (Section IV). (d) Extensive evaluation of proposed planning algorithms in realistic simulations driven by the measurements (Section V) to study performance in community/city-wide settings.

II. Spatiotemporal Scheduling in Hybrid Settings

In this section, we define frequently used terms and notations and model the system under appropriate assumptions. Based on the system model we formulate the spatiotemporal scheduling problem as a multi-objective optimization problem.

A. Notations and Assumptions

The area of interest (i.e. community/city) is discretized into **cells**. Cells can have arbitrary shapes, but we use square cells for simplicity. Cells are denoted by $c_i, i = 1, \dots, M$, where M is the total number of cells in the area of interest. The **spatial distance** between cells is represented by an M-by-M distance matrix $\mathbf{S}$, where S_{i_1,i_2} denotes the distance between the geometric centers of cells c_{i_1} and c_{i_2}.

A **data type** represents a class of sensor data we would like to collect in this area. Data types are written as $d_k, k = 1, \dots, K$, where K is the total number of data types of interest. We assume each data type requires a different type of sensors for collection. For simplicity, we also assume that we use the same type of sensor for the same data type. Hence there is a **one-to-one mapping**, where d_k also refers to its corresponding **sensor type**. The framework is extensible to complex settings where different sensors types are used for the same data type.

Each sensor type has its characteristics on temporal and spatial resolution. The temporal characteristic of d_k is captured by the **temporal impact function** $h_k^T(t)\in[0,1], t\geqslant 0$, which defines the contribution of a data point collected in the same cell, but time t ago. Similarly, the spatial characteristic of d_k is captured by the **spatial impact function** $h_k^S(s)\in[0,1], s\geqslant 0$, for contribution of a data point collected at the same time, but from a different cell that is distance s away. The selection of impact functions is application dependent; the framework should provide interfaces for applications to pass through their impact functions. Typical impact functions have some basic properties, such as $h(0)=1$ (full local impact), $h(x_1)\leqslant h(x_2)$ for $x_1<x_2$ (monotonicity), and $\lim_{x\to\infty} h(x)=0$. For example, in our environmental sensing scenario, we use exponential (spatial) and threshold-based (temporal) functions. Different types of sensors may generate data at different **rates**. Sensor of type d_k generates data at an average rate r_k. Each data type d_k is given an application-dependent **weight** p_k, s.t. $\sum p_k = 1$.

A **node** represents a participating device. It could be an in-situ sensing platform deployed in the community/city, or a crowd-owned mobile device roaming in the same area. A node is described by $\mathbf{n}_j, j = 1, \ldots, N$, where N is the total number of nodes. We assume we have **no control over the location or movement of nodes**. For consistency, we always use i, j, and k for indexes of cells, nodes, and data types, respectively. Each node has a subset of sensors **present** on-board. The presence of all sensor types on the nodes is represented by an N-by-K binary presence matrix $\mathbf{B}$, where $B_{j,k}=1$ iff $\mathbf{n}_j$ has the sensor of type d_k. In our hybrid settings, we do not differentiate between in-situ and mobile nodes explicitly in notations. Instead, we focus on node capabilities (i.e. sensors that are present on each node and their impacts).

The **placement** (i.e. location) of nodes is represented by an N-by-M binary placement matrix $\mathbf{G}(t)$, $t\geqslant 0$. $G_{j,i}(t)=1$ iff $\mathbf{n}_j$ is in c_i at time t. We assume that the placement of all the nodes is observable, i.e. $\mathbf{G}(t)$ is known at time t.

We assume each sensor on each node can be **activated** individually at any time. A **plan** describes which sensors on which nodes should be activated at what time. A plan is an N-by-K binary matrix $\mathbf{W}(t)$, where $t\geqslant 0$, $W_{j,k}(t)\leqslant B_{j,k}$, $W_{j,k}(t)=1$ iff d_k on $\mathbf{n}_j$ is active at t. A sensor generates data only when it is active. We say a node $\mathbf{n}_j$ is **active** at t, if at least one sensor on $\mathbf{n}_j$ is active at t.

In real world deployments, planning can occur at any time or when any change occurs; the associated data patterns generated and accumulated can vary. For simplicity, we assume a **discrete representation and operation** in our formulation, where time is discretized into time frames of length T, so that planning only occurs in intervals of $t = n{\cdot}T, \forall n\in\mathbf{N}$. The activation states of sensors, which is specified in the plan, persist throughout each time frame n. In this way, sensors can only be activated or deactivated per time frame. If an instance of d_k is active in one frame, then it generates $r_k{\cdot}T$ amount of data for that frame. In the discrete representation, we denote the discrete-time values by $\mathbf{G}[n]$, etc. where $\mathbf{G}[n]=\mathbf{G}(n{\cdot}T)$. We assume $\mathbf{G}(t)$ stays unchanged throughout any frame n.

B. Definition of Benefit, Cost and Constraints

Benefit: Our goal is to maintain up-to-date heatmaps. We evaluate the plan benefits for the collected data using two perspectives: spatiotemporal **coverage** and data **utility**. Coverage indicates how likely it is that a specific cell c_i has accurate data for data type d_k in time frame n, and utility indicates how useful those data items are, considering redundancy. Since both coverage and utility are closely related to the on/off state of sensors in each cell, we denote the activation state of d_k in c_i using an N-dimensional vector $\boldsymbol{\omega}_{i,k}[n] = [\,\omega_{i,1,k}[n], \ldots, \omega_{i,N,k}[n]\,]$, where $\omega_{i,j,k}[n] = W_{j,k}[n]{\cdot}G_{j,i}[n]$.

The **single-cell single-frame coverage** $\mathbf{X}^0[n]$ tells whether each cell is *directly* covered by data from at least one node. Its element is represented as

$$x^0_{i,k}[n] = x\big(\boldsymbol{\omega}_{i,k}[n]\big) = 1 - \prod_{j=1}^{N}\Big(1 - \omega_{i,j,k}[n]\Big). \qquad (1)$$

Even when a cell is not directly covered, it could have **effective coverage** from impacts of historical states and nearby cells. The single-cell single-frame effective coverage matrix $\mathbf{X}[n]$ has elements

$$x_{i,k}[n] = 1 - \prod_{\nu=0}^{n}\prod_{i'=1}^{M}\Big(1 - h_k^T(n-\nu){\cdot}h_k^S(S_{i,i'}){\cdot}x^0_{i',k}[\nu]\Big).$$

Therefore, the **spatial average coverage** in frame n, $x[n]$, is the average effective coverage over all data types in all cells, and $x[n'{:}n]$ is its average over time frames n' to n, i.e.

$$x[n] = \frac{1}{M}\cdot\sum_{k=1}^{K}\sum_{i=1}^{M} p_k{\cdot}x_{i,k}[n], \qquad (2)$$

$$x[n'{:}n] = \frac{1}{n-n'+1}\cdot\sum_{\nu=n'}^{n} x[\nu]. \qquad (3)$$

Similar to (1), the **single-cell utility** function can be written as $u_{i,k}[n]=u\big(\boldsymbol{\omega}_{i,k}[n]\big)$, where the selection of function u is application dependent. A general choice of function u should have these properties: (a) For any data type d_k in any cell, having data collected from more nodes in the cell is more useful than having data from only one node. (b) Having data from many different cells is globally more useful than having multiple data items from the same cell. Replacing letter x in Eq. (2) and (3), we get similar expressions for $u[n'{:}n]$.

Finally, the benefits, i.e. the **overall average coverage** X and the **overall average utility** U are derived respectively as

$$X = x[0{:}z], \;\; U = u[0{:}z], \qquad (4)$$

which we would like to maximize in our optimization, where z is the total number of time frames during entire operation.

Cost: We depict the cost of a plan using the **number of active nodes**, which reflects the overhead (e.g. core energy and user attention) to keep the nodes active. Scheduling policies can leverage this term to favor the situations where all sensors on some nodes are switched off. It also depends on plan $\mathbf{W}[n]$. According to our definition in Section II-A, we say $\mathbf{n}_j$ is active in time frame n if $\exists k$ s.t. $W_{j,k}[n] = 1$. Since $\mathbf{W}[n]$ is binary, that is equivalent to

$$y_j[n] = 1 - \prod_{k=1}^{K} \Big(1 - W_{j,k}[n]\Big),\ \ y[n] = \sum_{j=1}^{N} y_j[n], \qquad (5)$$

where $y_j[n]$ is the single-frame activation state of node $\mathbf{n}_j$ and $y[n]$ is the total number of active nodes in time frame n. The average node activation over multiple time frames $y[n'{:}n]$ is written by replacing the x letter in Eq. (3) with y, i.e. the **average number of active nodes** $Y = y[0{:}z]$, which we use as cost to minimize in our optimization.

Constraints: One key constraint that we capture by our definition is $W_{j,k}[n] \leqslant B_{j,k}, \forall j = 1,\dots,N, \forall k = 1,\dots,K, \forall n \in \mathbf{N}$, which reflects the **hardware configuration**, i.e. no device could activate a sensor that does not exist on it.

The other constraint is the **data quota** determined by the limited server resources and communication infrastructure so that all data could be transferred and processed timely. The total data rate in frame n is the number of active sensors of each type multiplied by the type-specific data rate, i.e. $d[n] = \sum_{k=1}^{K} r_k \cdot \sum_{j=1}^{N} W_{j,k}[n]$, and the average data generation rate through time frames n' to n is $d[n'{:}n] = \sum_{\nu=n'}^{n} d[n]/(n - n' + 1)$. Thus, the **average data rate** $D = d[0{:}z]$. Note the dimension of D is *byte/s*. With our optimization, we would like to keep D bounded within a predefined data quota D_{quota}.

C. Problem Formulation

With the assumptions and terms we have, we formulate the spatiotemporal scheduling problem as the following multi-objective optimization problem:

In any time frame $n \in \mathbf{N}$, given the sensor type presence matrix $\mathbf{B}$, the nodal placement matrix $\mathbf{G}[n]$ and the data type characteristics r_k, h_k^T, and h_k^S, $k = 1,\dots,K$, determine $\mathbf{W}[n]$ that optimizes the expectation of **overall performance** $\mathbf{E}[\Gamma]$, which is defined as the weighted sum of (a) the average coverage X, (b) the average utility U, and (c) the average number of active nodes Y, subject to the hardware configuration and data quota constraints. Formally, this is stated as

$$\begin{aligned} \max_{\mathbf{W}[n]} \mathbf{E}[\Gamma(X,U,Y)] &= \gamma_1 \cdot \mathbf{E}[X] + \gamma_2 \cdot \mathbf{E}[U] - \gamma_3 \cdot \mathbf{E}[Y], \quad (6) \\ \text{s.t. } & W_{j,k}[n] \leqslant B_{j,k}, \forall j = 1,\dots,N, \forall k = 1,\dots,K, \\ & \mathbf{E}[D] \leqslant D_{\text{quota}}. \end{aligned}$$

This formulation of the spatiotemporal scheduling problem, even when simplified into its single time frame case used in our online approach, is a typical integer programming problem which is known to be NP-hard.

III. Algorithms for Online Scheduling

In the absence of a scheduling technique, a simplistic policy is that of complete activation (i.e. activating every available sensor). This naïve "everything" approach may not meet data constraints, but always results in the maximum possible overall coverage and utility with the given inputs and can be used for comparison purposes in evaluation. A brute-force search will compute the optimal solution but obviously only for very small test cases. In this section, we propose 2 algorithms to address the online scheduling problem: (a) an iterative greedy heuristic with improved algorithm termination (Sec. III-A), and (b) a Lyapunov control strategy inspired optimization (Sec. III-B).

Algorithm 1: Highest-score-first algorithm for finding a plan $\mathbf{W}[n]$ for time frame n, showing the basic procedure and the slow termination phase

```
1  function planHSF ($\mathbf{B}$, $\mathbf{G}[n]$, $D_q$, $\{d\}$, $\mathbf{S}$, $\boldsymbol{\gamma}$, $\mathbf{X}^0$) ;
   Input : Presence $\mathbf{B}$, placement $\mathbf{G}[n]$, and quota $D_q$
           Data types $\{d\}$ and their characteristics
           Spatial distance $\mathbf{S}$ and weights of objectives $\boldsymbol{\gamma}$
           Historical single-cell coverage $\mathbf{X}^0[0:(n-1)]$
   Output: Plan $\mathbf{W}[n]$ for time frame $n$
2  Initialize $\mathbf{W}[n] \leftarrow \mathbf{0}_{N,K}$ ; $j_{ST} \leftarrow$ null;
3  $\mathbf{X}^0[n] \leftarrow$ Get single-cell coverage from $\mathbf{W}[0:n]$ ;
4  cand $\leftarrow \{(j,k) \mid B_{j,k} = 1 \wedge W_{j,k}[n] = 0\}$ ; sumD $\leftarrow 0$ ;
5  while cand is not empty do
6      maxScr $\leftarrow 0$ ; maxPr $\leftarrow$ null;
7      for each $(j,k)$ in cand do
8          $\mathbf{W}' \leftarrow \mathbf{W}$ ; $W'_{j,k} \leftarrow 1$ ;
9          $\delta_{j,k} \leftarrow$ Get score from $\mathbf{W}[0:(n-1)]$ and $\mathbf{W}'$ ;
10         if $\delta j,k >$ maxScr and sumD $+ r_k \leqslant D_q$ then
11             maxScr $\leftarrow \delta_{j,k}$ ; maxPr $\leftarrow (j,k)$ ;
12     if maxPr is not null then
13         if $j_{ST}$ is null then $W_{\text{maxPr}}[n] \leftarrow 1$ ;
14         else
15             $w_{\text{ST},k} \leftarrow 1$ ; sumScr $\leftarrow$ sumScr $+ \delta_{j,k}$ ;
16             if sumScr $> 0$ then
17                 $j_{\text{ST}} \leftarrow$ null; $W_{j_{\text{ST}},*} \leftarrow \mathbf{w}_{\text{ST}}$ ;
18         sumD $\leftarrow$ sumD $+ r_k$ ;
19     else if $j_{ST}$ is null then
20         $(j,k) \leftarrow$ Get pair for max score without $\Delta y$ ;
21         $j_{\text{ST}} \leftarrow j$ ; $\mathbf{w}_{\text{ST}} \leftarrow \mathbf{0}_K$ ; sumScr $\leftarrow \delta_{j,k}$ ;
22         maxPr $\leftarrow (j,k)$ ; sumD $\leftarrow$ sumD $+ r_k$ ;
23     if maxPr is not null then cand.delete maxPr ;
24     else break ;
```

A. Highest-Score-First Greedy Heuristic

The **highest-score-first (HSF)** greedy heuristic computes a *score* for each sensor on each node (i.e. each node-sensor pair) in a time frame n. It iteratively chooses the node-sensor pair with the highest score for activation, and updates the scores for other pairs, until no selection yields a positive score. The data quota can roll over to the next time frame but cannot be advanced. Its balance in current frame is denoted by D_q Algorithm 1 sketches the overall technique.

The **score** $\delta_{j,k}$ of a node-sensor pair (j,k) is defined as the unit-data contribution it makes to the overall performance if we activate (only) this sensor, i.e. $\delta_{j,k} = p_k \cdot (\gamma_1 \cdot \Delta x[n] + \gamma_2 \cdot \Delta u[n] - \gamma_3 \cdot \Delta y[n])/\Delta d[n]$, where $\Delta d[n] = r_k$. The intuition behind HSF is to enhance the overall performance metric $\mathbf{E}[\Gamma]$ in Equation 6 using the limited data quota.

HSF starts with an empty matrix $\mathbf{W}[n]$, where all sensors are inactive, i.e. $W_{j,k}[n] = 0, \forall j = 1, \ldots, N, \forall k = 1, \ldots, K$. In every iteration (big loop, Ln 5–24), it computes the score $\delta_{j,k}$ for each (j, k) pair, s.t. $B_{j,k}$=1 and $W_{j,k}[n]$=0 (sensor k exists on node j but not yet activated), and picks the (j, k) that gives the maximum positive $\delta_{j,k}$. To compute the scores of all pairs (inner loop, Ln 7–11), for each pair (j_0, k_0), it creates a temporary plan W', s.t. $W'_{j_0,k_0} = 1$ and $W'_{j,k} = W_{j,k}[n], \forall (j, k) \neq (j_0, k_0)$, computes the objective values of W', and subtract that of $W[n]$ from the result to acquire the score δ_{j_0,k_0}. HSF loops until no positive score is possible, or the data quota is used up.

Complexity: The intuitive implementation of HSF has worst-case time complexity of $O(K^2M^2N^2 + K^2MN^3)$, which can be reduced to $O(KM^2N + KMN^2 + K^2N^2)$ with appropriate optimization.

Early termination of HSF: Occasionally HSF terminates due to non-positive scores; if we select a pair (j_0, k_0) with non-positive score, succeeding selections can be made on the same node to amortize the cost of activating j_0, so all active sensors on j_0 together can make a positive total score. In this case, j_0 is a node that should be included in $W[n]$, but were not because of HSF's early termination.

Therefore, we add a "slow termination" (ST) phase depicted in Alg. 1, Ln 12–22 to HSF. In **HSF–ST**, when the scheduler meets all non-positive scores, it moves (using Ln 19–22) into the slow termination phase (marked by node j_{ST}). Specifically in this phase, it picks the (j, k) that gives the highest score when not considering the $\Delta y[n]$ term in score computation (Ln 20). Then it iteratively adds other sensors on the same node j, keeping track of the total score of all added sensors on node j (Ln 14–17). There are three possible ways that ST ends: (a) After adding a sensor, the accumulative score turns positive (Ln 16). In this case, it adds all sensors selected during ST into $\mathbf{W}[n]$ (Ln 17) and goes back to the basic HSF loops. (b) Data quota is used up. (c) All sensors on node j are selected, but the accumulative score is still non-positive. In both (b) and (c), all sensors selected during ST are dropped, and $\mathbf{W}[n]$ is returned immediately (Ln 24).

Complexity: The worst-case time complexity of HSF–ST is the same with basic HSF, i.e. $O(KM^2N + KMN^2 + K^2N^2)$.

B. Data Budget Handling using Lyapunov Control

In HSF–ST (Section III-A), we try to optimize the overall performance up to time frame n, without considering future possibilities. However, on occasion, we may want to save data for future use when present benefits are marginal, or advance data quota to seize the opportunity for a significant improvement. The Lyapunov control strategy allows us to dynamically handle data budget while keeping the average usage bounded.

Lyapunov optimization [13] is often applied to systems that evolve over time. It maximizes the temporal average reward while keeping one or more "queues" bounded. The framework defines a Lyapunov function on system states and tries to keep the Lyapunov drift (expected difference between function values at two successive steps), as small as possible, which ultimately ensures the system reaches its goal over time.

In our spatiotemporal scheduling, we define the system state $\phi[n]$ as a queue representing the accumulative data rate, i.e. $Q[n] = n \cdot d[0{:}(n-1)]$, which equals the amount of data that have been generated up to frame $(n-1)$ divided by frame length T. Then, the Lyapunov function is $L(\phi[n]) = (Q[n]-n\cdot D_{\text{quota}})^2/2$, and the Lyapunov drift is $\Delta L(\phi[n]) = \mathbf{E}[L(\phi[n+1]) - L(\phi[n])] \approx (Q[n] - n \cdot D_{\text{quota}}) \cdot d[n]$, where $d[n] = Q[n+1] - Q[n]$. The strategy suggests we minimize the "drift minus reward" $(\Delta L(\phi[n]) - V \cdot R[n])$, which is to maximize

$$\gamma_1 \cdot x[0{:}\, n] + \gamma_2 \cdot u[0{:}n] - \gamma_3 \cdot y[0{:}n] - \gamma_L \cdot \Delta L(\phi[n]). \quad (7)$$

In the actual implementation, we use the same γ values as coefficients for X, U, and Y; we pick a value in the order of 10^{-9} for γ_L. Tuning of γ_L and a few other minor tweaks are needed to establish consistency when the scenario scales.

Complexity: The complexity of Lyapunov control depends on the implementation of the maximizer. In our case, we employ a similar greedy heuristic as is used in Algorithm 1 to maximize Eq. 7, using the increment of its value as the score in selection of node-sensor pairs. It has the same complexity with HSF–ST.

IV. Platform and Measurements

The spatiotemporal scheduling approach we propose is derived from our existing community-oriented IoT deployments in four different locations: (a) UCI campus in Irvine, CA, (b) the Thingstitute lab and Victory Court Senior Apartments in Montgomery County, MD, (c) NTHU campus in Hsinchu City, Taiwan, and (d) Dhaka, Bangladesh. To determine the feasibility of our approach and collect measurements to drive our simulations, we created a prototype platform and carried out measurement studies using varying and realistic combinations of sensors in three testbeds. Collected data provides data generation patterns of several common application/sensor types in urban crowd-sensing scenarios.

The Flexible Sensing Platform: We leverage SCALE2 [14] and SCALECycle [15], our existing community IoT projects to design the sensing platform that consists of three major components: (a) the end devices (insitu or mobile) (b) a community server that monitors and manages the devices, and (c) the data exchange service that supports their communication (Fig. 2a). End devices are prototyped using Raspberry Pi 2 Model B with necessary components, including Wi-Fi adapter, ADC board, power supply, and optional microphone (USB) and camera (CSI), etc. Analog sensors include several MQ family sensors (MQ–5,7,131,135) and TGS 2600. A mobile node is powered on a USB battery pack, and has a GPS module (USB or Bluetooth). An in-situ node gets power from a USB wall charger. The Raspbian-based RPi platform runs a client middleware (implemented in Python), which consists of a local message broker and multiple applications that communicate with each other. Applications include virtual sensors (data sources), publishers (data sinks), network manager, etc. When

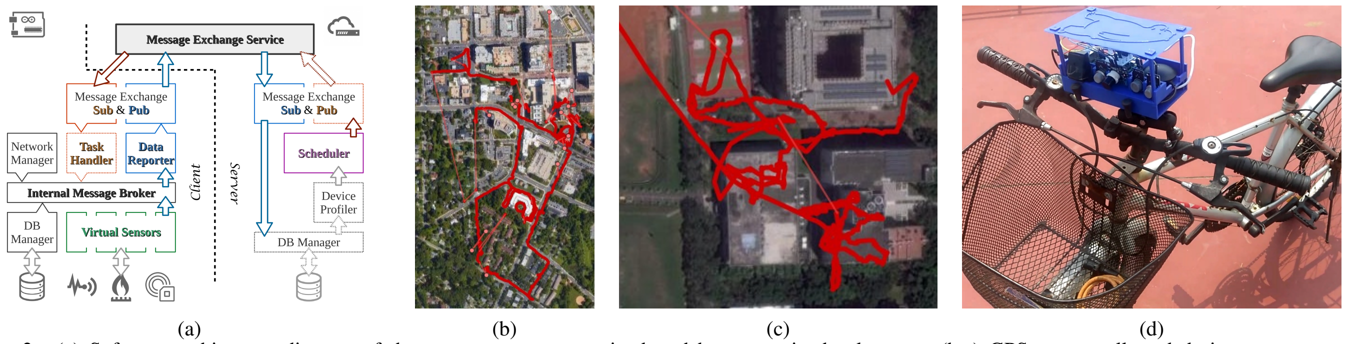


Fig. 2. (a) Software architecture diagram of the system, components in dotted boxes are in development. (b,c) GPS traces collected during two runs of measurement study on the Montgomery County and the NTHU testbeds, respectively. (d) Photo of a mobile node mounted on a bicycle.

TABLE I
DATA GENERATION PATTERN USED IN MEASUREMENT STUDY.

Sensor Type	Format	Sampling Pattern	Sample Size
Gas (Analog)	JSON	1 message every 5 sec	200
Microphone	WAV	Clip of 8 sec every min	800k
Camera	JPEG	1 picture every 20 sec	180k
Wi-Fi (iw)	JSON	List of 20 items every 4 sec	3.6k

connected to the Internet, the collected data is delivered to a data exchange service using the MQTT protocol [16]. The mobile node can be mounted on bikes as is shown in Fig. 2d. The server is a desktop PC that uses data exchange services to receive end device information and publish commands (task assignments). The data exchange service is currently implemented by a desktop PC that runs the Mosquitto MQTT broker, but can be replaced by any MQTT-enabled broker. Both PCs are connected to our institution's LAN using a 100 Mbps Ethernet network.

Initial Measurements on Real Testbeds: We collected initial measurements using the prototype end device with several different hardware configurations on **three of our real-world testbeds**: On the UCI and the Montgomery County testbeds, the device is equipped with a TGS 2600 air contaminant sensor and uses its Wi-Fi adapter to collect RSSI data for nearby Wi-Fi APs. On the NTHU testbed, one profile has four MQ sensors and continuous collection of air pollution data at a sampling rate of 0.2 Hz. Another one has a USB microphone and a CSI camera module. Fig. 2(b,c) shows the GPS trace collected during two runs of measurement study. Data generation patterns we observed are listed in Table I. The sample size shown in the table is for uncompressed raw data. Different application/data types exhibit unique patterns and rates of data generation.

V. PERFORMANCE EVALUATION

In this section, we describe the simulation environment and experimental design for performance evaluation, and present the results with analyses.

A. Experimental Environment and Simulation Setup

Earlier in Section IV, we built prototype devices to demonstrate the effectiveness of our system architecture and conduct

TABLE II
BASIC SIMULATION SETUP.

Simulation Parameters		Values
Cells	Number M	63
	Size (Length) l_{cell}/m	50
Nodes	Number of All N	100 *or* 50 [a,b]
	Number of Mobile Nodes N_{mob}	40 *or* 20 [a,b]
	Sensor Presence	See Table III [b]
	Speed of Mobile Nodes $v/(\text{m/s})$	$[0.5, 1.5]$ [a]
Data Types	Number K	10
	Characteristics	See Table III
Simulation	Length of Time Frame T/min	1
	Length of Simulation $T_{\text{dur}}/\text{min}$	180
	Data Quota $D_{\text{quota}}/(\text{MB/s})$	3.5 [a]
	Weights of Objectives $\boldsymbol{\gamma}$	$(1, 0.4/\ln 2, 1.2)^t$

[a] Subject to change if used as an independent variable.
[b] Setup varies in simulation scenarios.

measurement studies on real-world testbeds. Due to the limited scale of our current deployments, we carry out further evaluation in simulations driven by measurements from our testbeds. Simulations are performed using the ONE simulator [17] and our custom simulation framework written in R. Data generation rates and patterns are tuned to reflect our real-world measurements; and mobility traces are generated by the ONE simulator using its pedestrian model on the built-in Helsinki street map [17]. The performance evaluation component and scheduler interface are implemented by our custom simulator written in R, where we can add algorithms as R functions. Our simulations do not consider communication delays or faults. Experiments were done on the ICS Openlab cluster at UCI, where each node has two (2x) Quad-core Intel Xeon 3.0 GHz CPUs, 32 GB RAM, and runs CentOS 7.3.

The basic experimental setup is shown in Table II. As is assumed in Sec. II-A, we use nodal locations at the beginning of frame n as the prediction of $\mathbf{G}[n]$ which stays the same throughout the frame n, i.e. $\mathbf{G}[n] = \mathbf{G}(n \cdot T)$. This assumption holds as $v \cdot T \ll l_{\text{cell}}$. The simulation framework is extensible to support complex predictors. We define the spatial impact h_k^S for each individual data type using a exponential function $h_k^S(s) = \exp(-s/S_k^0), s \geqslant 0$ with a type-specific constant $S_k^0 \geqslant 0$. We define the temporal impact h_k^T using a step-down function $h_k^T(t) = 1 - \theta(t - T_k^0), t \geqslant 0$ with a type-specific

TABLE III
DATE TYPE CHARACTERISTICS AND SENSOR PRESENCE ON NODES.

Data Type	Model	Rate	Wt.	Impact		Presence [a]	
		r_k (B/s)	p_k	S_k^0 (m)	T_k^0 (min)	P_{mob}	P_{sta}
d_1–d_6	Analog	40	– [b]	500+	20	– [c]	1
d_7	Audio	107k	0.10	100	5	0.6	1
d_8	Picture	9k	0.14	10	60	0.9	0
d_9			0.12	750	30	0.3	0.7
d_{10}	Wi-Fi	900	0.06	60	20	1	1

[a] Represented by probability of presence on mobile and static nodes.
[b] Values of p_1–p_6 are respectively 0.16, 0.10, 0.10, 0.08, 0.08, and 0.06.
[c] Each mobile node has four out of six analog gas sensors on-board.

constant $T_k^0 \geqslant 0$, where θ denotes the Heaviside step function. We use the logarithm utility function $u(\boldsymbol{\omega}) = \log(1 + \sum \boldsymbol{\omega})$. Data type specs are shown in Table III.

We acquired baselines from the naïve **everything** approach and a utility-driven **random greedy algorithm** that chooses sensors from cells where highest utility gain can be achieved until data quota is met, which degrades to "everything" with infinite quota. We compared them with HSF–ST and Lyapunov control algorithms. Test sets are designed to reveal the impact of (a) node mobility, (b) device heterogeneity, and (c) scale.

We also tested a basic genetic algorithm (GA) [18] with a limit on running time set to frame length T=1 min (Fig. 4c). GA did not appear to fit in online scheduling and performed badly in most of the tests (e.g. Fig. 4a). We believe this is because the time frame is too tight for GA to converge as the solution space expands exponentially when system scales up. Thus, GA is excluded from most of our results.

B. Performance Evaluation Metrics

1) Overall Performance: The **overall performance** is the optimization goal, given by the weighted sum of objectives as is formulated in Equation 6. Note that the overall performance considers both **benefit** and **cost**.

2) Coverage, Utility, and Number of Active Nodes: These are individual objectives defined in Section II-B. We compare our approaches with them for in-depth analyses. Number of active nodes is normalized by $N \cdot K$.

3) Data Generation and Quota Utilization: According to our formulation in Section II-B, the operation is constrained by the data quota. Satisfying the long-term data quota constraint, a good algorithm usually maximizes its quota utilization.

4) Scheduling time: Scheduling time is the running time of the scheduling algorithm. In our proposed online approach, all computation needs to be done on the server within a limited time (shorter than the time frame). Hence, it is important to compare the running time of the algorithms.

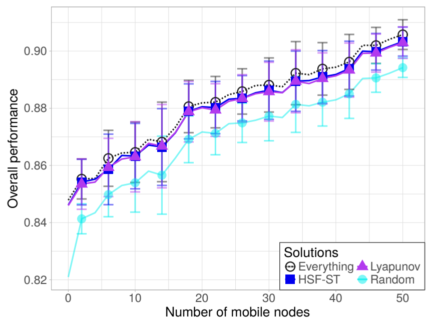
C. Simulation Results

1) Impact of Node Mobility: Fig. 3(a,b,c) display the results on varying node mobility. In Fig. 3a and 3b, the total number of nodes is fixed at N=50. As the number of mobile nodes among them increases, we notice that the overall performance increases for all tested algorithms (Fig. 3a), while the average number of active nodes stays almost unchanged (Fig. 3b). Similar trend has been observed when we increase the moving speed of the pedestrian model (Fig. 3c, all nodes are mobile). These results show the benefits of using mobile sensing nodes as augmentation to existing in-situ deployments. In all these tests, HSF–ST and Lyapunov control show superior performance over the random greedy algorithm. Both our algorithms achieve overall performance close to "everything", while using 25–30% fewer active nodes.

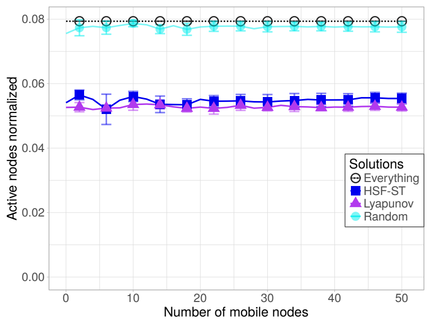
2) Impact of Device Heterogeneity: Fig. 3(d,e,f) displays the results on varying device heterogeneity. Device heterogeneity is represented by the distribution of a set of sensors of each type across all nodes; all nodes are mobile in this test set. With more mobile nodes, the sensors are distributed more sparsely, so more nodes need to be activated (Fig. 3f, especially the "everything" and "random" curve) to achieve the same level of coverage (Fig. 3e). This means lower overall performance, because the overall performance considers the cost of node activation. In comparison, HSF–ST and Lyapunov control achieve better overall performance by balancing benefit (coverage and utility) and cost (node activation). The improvement is about 10% for N=200; this increases as the deployment scales up. Lyapunov control performs better than HSF–ST, especially when node heterogeneity increases, likely because the non-uniform distribution of sensors could trigger significant benefits that need to be balanced over time.

3) Impact of Scale: Fig. 4 displays results for the impact of scale. Fig. 4(a,b,c) are results for experiments where we tune the total number of nodes, with a fixed 40% of nodes being mobile. As the total number of nodes increases, the marginal performance becomes trivial and gradually gets overwhelmed by the penalty from node activation (Fig. 4a, the "everything" curve). The random greedy algorithm does not scale, and its performance gets worse for $N \geqslant 40$. In comparison, HSF–ST and Lyapunov control achieve better-than-"everything" overall performance by nicely balancing benefits and cost. Fig. 4b shows HSF–ST and Lyapunov control makes full use of the data quota without violation. Fig. 4c shows a polynomial increase in scheduling times as predicted when the total number of nodes is small, and linear growth when the number of nodes approaches the "appropriate" value for the map. In this test set, HSF–ST seems to be slightly better than Lyapunov control, especially when the number of nodes is big. HSF–ST also runs about 25% faster than Lyapunov control in our settings. A trade-off between spatial and temporal resolution can be inferred here, i.e. a longer time frame should allow planning for more nodes on a large map with more cells.

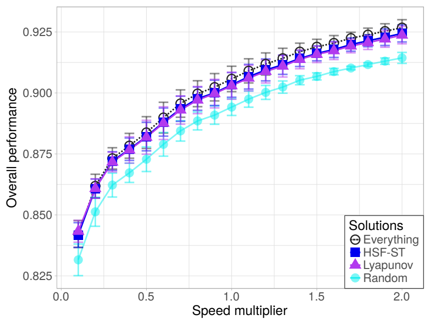
Fig. 4d is for a similar test set, but the number of in-situ nodes is fixed at 30 while the number of mobile nodes grows from 0 to 170. Fig. 4e and 4f show performance and quota utilization when data quota increases from a very small value (hundreds of kilobytes) until it is more than sufficient. We find that Lyapunov control achieves better overall performance and quota utilization when data quota is larger. Both algorithms result in up-to 15% better overall performance as compared to the random greedy algorithm.



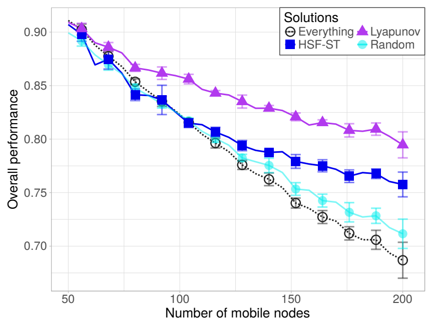
(a) Overall performance vs. node mobility, using fixed total number of nodes, $N=50$



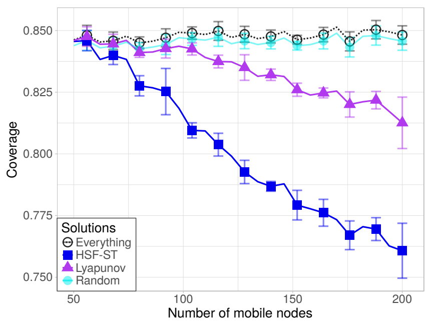
(b) Active nodes vs. node mobility, $N=50$



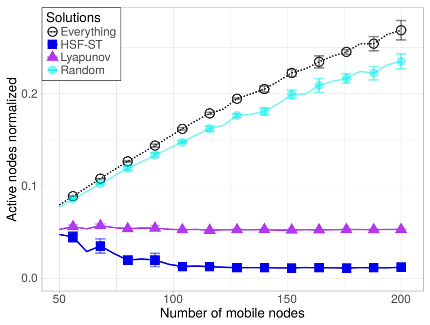
(c) Overall performance vs. speed of mobile nodes (multiplier times $v \in [0.5, 1.5]$ m/s), $N_{\text{mob}}=50$



(d) Overall performance vs. device heterogeneity



(e) Coverage vs. device heterogeneity



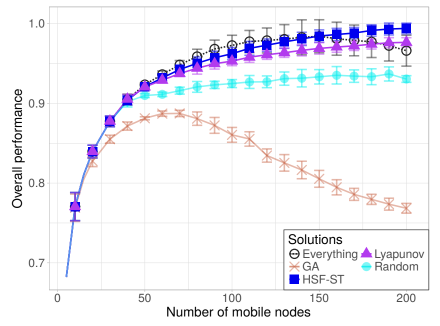
(f) Active nodes vs. device heterogeneity

Fig. 3. Simulation results for impact of node mobility and device heterogeneity.

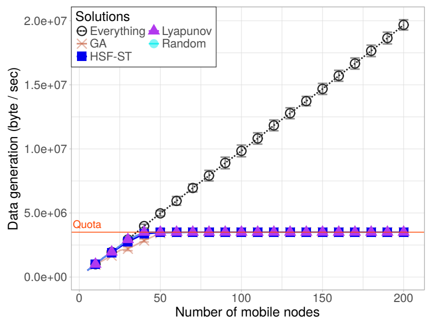
VI. Related Work and Conclusion

Resources and data from crowd users have been leveraged in applications [19] including voting systems, information sharing systems, and social games. In recent years, smartphones with sensors have been used to engage crowd user participation in spatiotemporal dependent tasks. Kanhere [20] points out several challenges in participatory sensing, such as context-awareness and energy conservation. Crowd incentives are also crucial when leveraging crowd resources, and most studies employ monetary incentives [21], [22]. For example, Feng et al. [21] present an auction framework for the crowd with smartphones in order to maintain truthfulness and individual rationale. Incentives other than monetary have been studied in more recent work [23]–[25]. Talasila et al. [24] and Chen et al. [25] propose to leverage mobile and even augmented-reality games to transparently guide mobile gamers to certain places to perform sensing tasks. Our work concentrates on the spatiotemporal scheduling problem, and is orthogonal to the aforementioned related work. Benson et al. [26] and Uddin et al. [14] create crowd sensing systems for safety awareness in communities. Liao et al. [6] propose a platform that combines crowd and in-situ sensors for urban sensing. Their work only considers individual tasks at discrete locations; in contrast, our work strives to build complete sensor reading maps in real-time. Zhu et al. [15] propose to leverage node mobility for better coverage and timely data collection in communities. Han et al. [27] formulate a utility maximization framework for mobile crowd sensing that balances data utility and incentive. Khan et al. [28] build a localization framework to estimate the block-level location of participating mobile devices to lower the usage of GPS for energy conservation. Marjovi et al. [1] and Hasenfratz et al. [2] propose to leverage mobile entities in cities to help create high-resolution pollution maps, and focus on using data from a small group of devices and leveraging offline machine learning based techniques to infer the states in uncovered areas. Our work mainly considers device and data heterogeneity and stresses on sensor activation strategies to balance coverage and redundancy. Hachem et al. [29] build a registration middleware for city-scale participatory sensing systems to reduce participation based on predicted probability of path and capability overlap. Our approach applies to real-time monitoring applications and dynamically selects sensors to activate.

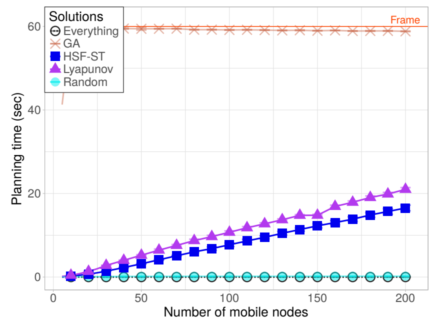
In this paper, we motivated and formalized the spatiotemporal scheduling problem in crowd augmented urban sensing systems. We proposed an online scheduling framework and two scheduling algorithms that address the challenges of heterogeneity and scalability. We developed prototype platforms, deployed them in three real community testbeds and collected measurements that were then used to drive extensive simulations. Experimental results showed that, in comparison to naïve approaches, the proposed algorithms (HSF–ST and Lyapunov) are significantly more efficient and scalable in heterogeneous community/city settings. Future work aims at further addressing scalability issues through the use of scheduling hierarchies to offload work to edge servers and devices [30]. We also plan to explore the impact of uncertainties in network connectivity and node mobility on plan execution using different mobility prediction models. Such research is a key enabler to engaging



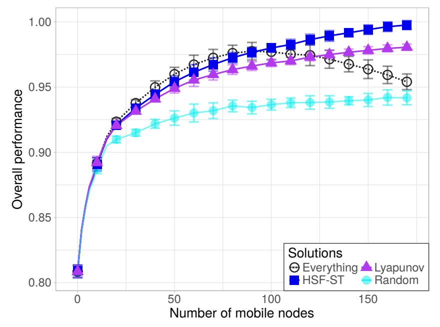
(a) Overall performance vs. total number of nodes



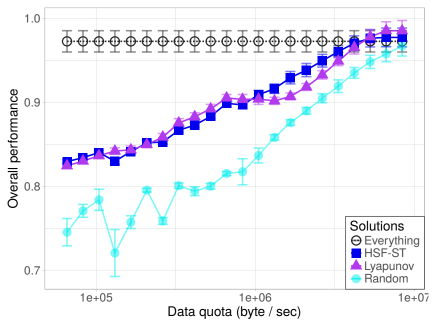
(b) Data generation vs. total number of nodes



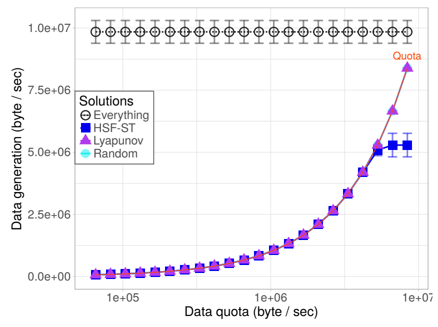
(c) Scheduling time vs. total number of nodes



(d) Overall performance vs. num. of mobile nodes



(e) Overall performance vs. data quota



(f) Data generation vs. data quota

Fig. 4. Simulation results for impact of scale.

human participation in smart community deployments.

Acknowledgment: The authors thank all members of the ISG and DSM groups at UCI for the valuable discussions, and the volunteers for their help with measurement study. This project is funded under NSF CNS-1450768 and NIST MSE 446677-22553.

References

[1] A. Marjovi, A. Arfire *et al.*, "High Resolution Air Pollution Maps in Urban Environments Using Mobile Sensor Networks," in *DCOSS '15*.
[2] D. Hasenfratz, O. Saukh *et al.*, "Pushing the spatio-temporal resolution limit of urban air pollution maps," in *PerCom '14*, pp. 69–77.
[3] S. Devarakonda, P. Sevusu *et al.*, "Real-time Air Quality Monitoring Through Mobile Sensing in Metropolitan Areas," in *SIGKDD '13 Workshop on Urban Computing*, ser. UrbComp '13, pp. 15:1–15:8.
[4] M. AbuJayyab, S. A. Ahdab *et al.*, "PolluMap: A Pollution Mapper for Cities," in *2006 Innovations in Information Technology*, Nov. 2006.
[5] Y. Zheng, T. Liu *et al.*, "Diagnosing New York City's Noises with Ubiquitous Data," in *UbiComp '14*, pp. 715–725.
[6] C. Liao, T. Hou *et al.*, "SAIS: Smartphone augmented infrastructure sensing for public safety and sustainability in smart cities," in *EMASC '14*, pp. 3–8.
[7] R. K. Ganti, F. Ye *et al.*, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, 2011.
[8] R. Beelen, O. Raaschou-Nielsen *et al.*, "Effects of long-term exposure to air pollution on natural-cause mortality: an analysis of 22 european cohorts within the multicentre escape project," *The Lancet*, 2014.
[9] N. Künzli, R. Kaiser *et al.*, "Public-health impact of outdoor and traffic-related air pollution: a european assessment," *The Lancet*, 2000.
[10] R. T. Burnett, J. Brook *et al.*, "Association between particulate-and gas-phase components of urban air pollution and daily mortality in eight canadian cities," *Inhalation toxicology*, 2000.
[11] X. Xu, J. Gao *et al.*, "Air pollution and daily mortality in residential areas of beijing, china," *Archives of Environmental Health: An International Journal*, vol. 49, no. 4, pp. 216–222, 1994.
[12] J. Zhang, Y. Sun *et al.*, "Characterization of submicron aerosols during a month of serious pollution in beijing, 2013," *Atmospheric Chemistry and Physics*, vol. 14, no. 6, pp. 2887–2903, 2014.
[13] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.
[14] M. Y. S. Uddin, A. Nelson *et al.*, "The SCALE2 multi-network architecture for iot-based resilient communities," in *SMARTCOMP '16*.
[15] Q. Zhu, M. Y. S. Uddin *et al.*, "Upload planning for mobile data collection in smart community internet-of-things deployments," in *SMARTCOMP '16*, pp. 1–8.
[16] MQTT. [Online]. Available: http://mqtt.org/
[17] A. Keränen, J. Ott *et al.*, "The ONE simulator for DTN protocol evaluation," in *ICST '09*, p. 55.
[18] L. Scrucca, "GA: A package for genetic algorithms in R," *Journal of Statistical Software*, vol. 53, no. 4, pp. 1–37, 2013.
[19] M. Yuen, I. King *et al.*, "A survey of crowdsourcing systems," in *PASSAT '11 and SocialCom '11*, pp. 766–773.
[20] S. Kanhere, "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces," in *MDM '11*, 2011, pp. 3–6.
[21] Z. Feng, Y. Zhu *et al.*, "TRAC: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing," in *INFOCOM '14*.
[22] M. Talasila, R. Curtmola *et al.*, "Crowdsensing in the wild with aliens and micropayments," *IEEE Pervasive Computing*, vol. 15, Jan 2016.
[23] F. Alt, A. Shirazi *et al.*, "Location-based crowdsourcing: Extending crowdsourcing to the real world," in *NordiCHI '10*, 2010, pp. 13–22.
[24] M. Talasila, R. Curtmola *et al.*, "Alien vs. mobile user game: Fast and efficient area coverage in crowdsensing," in *MobiCASE '14*, pp. 65–74.
[25] Y. Chen, H. Hong *et al.*, "Gamifying mobile applications for smartphone augmented infrastructure sensing," in *NetGames '17*, pp. 12:1–12:16.
[26] K. Benson, C. Fracchia *et al.*, "SCALE: Safe community awareness and alerting leveraging the internet of things," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 27–34, 2015.
[27] Y. Han, Y. Zhu *et al.*, "Utility-maximizing data collection in crowd sensing: An optimal scheduling approach," in *SECON '15*, pp. 345–353.
[28] A. Khan, S. K. A. Imon *et al.*, "A novel localization and coverage framework for real-time participatory urban monitoring," *Pervasive and Mobile Computing*, vol. 23, pp. 122–138, 2015.
[29] S. Hachem, A. Pathak *et al.*, "Probabilistic registration for large-scale mobile participatory sensing," in *PerCom '13*, pp. 132–140.
[30] H. Hong, P. Tsai *et al.*, "Supporting internet-of-things analytics in a fog computing platform," in *CloudCom '17*.
[31] Array of things. [Online]. Available: medium.com/array-of-things