Energy-Efficient Event Detection by Participatory Sensing Under Budget Constraints

Chi Harold Liu, *Senior Member, IEEE*, Jianxin Zhao, Honggang Zhang, *Senior Member, IEEE*, Song Guo, *Senior Member, IEEE*, Kin K. Leung, *Fellow, IEEE*, and Jon Crowcroft, *Fellow, IEEE*

***Abstract*—Dynamic event detection by using participatory sensing paradigms has received growing interests recently, where detection tasks are assigned to smart-device users who can potentially collect needed sensory data from device-equipped sensors. Typical applications include, but are not limited to, noise and air pollution detections, people gathering, even disaster prediction. Given this problem, although many existing centralized solutions are effective and widely used, they usually cause heavy communication overhead. Thus, it is strongly desired to design distributed solutions to reduce energy consumption, while achieving a high level of detection accuracy with limited sensing task budget. In this paper, we first present two novel centralized detection algorithms as the performance benchmark, which make use of the *Minimum Cut* theory and support vector machine (SVM)-based pattern recognition techniques. Then, we introduce a novel distributed and energy-efficient event detection framework under task budget constraint, where we formulate an optimization problem and derive an optimal utility function. Finally, based on a real trace-driven data set in an urban area of Beijing, extensive simulation results demonstrate the effectiveness of our proposed algorithms.**

***Index Terms*—Distributed event detection, energy efficiency, incentive budget, participatory sensing.**

I. Introduction

PARTICIPATORY sensing is an emerging paradigm that aims at collecting useful data from a large number of surrounding smart-device users. These smart devices are usually equipped with a variety of sensors, e.g., accelerometer, gyroscope, GPS, light sensor, microphone, and camera, that can monitor almost every aspect of our daily surrounding environment. Participatory sensing can successfully reduce the infrastructure deployment costs and enable us to sense the world at an unprecedented spatiotemporal granularity. These features make it quite suitable to be applied in a wide range of applications, e.g., monitoring of noise level [2], air quality [3], and street-parking availability [4] in urban areas.

Among many applications of participatory sensing, dynamic event detection has received a growing amount of research attention. Specifically, our focus in this paper is motivated by an application scenario as follows. A group of smart-device users move inside a certain spatial sensing region, while subscribing to a central server (CS). Periodically, a selected crowd of participants collect sensory data by using their smart-device-embedded sensors. Sometimes, local data processing is also needed. These participants then transmit necessary data to the CS via built-in communication interfaces such as 3G/LTE or WiFi. By processing these data, the CS is able to identify target event areas inside the sensing region with certain detection accuracy. Such events could be abnormal noise level in a specific region, residential fire in forests, or meteorological hazards [5]–[7], etc.

There are mainly two types of technical approaches to enable the running of the aforementioned application scenario. First is the *centralized* approaches, where participants transmit the collected raw data to the CS for further processing [7] (it is worth noting that the processing can also take place in a distributed cluster, e.g., Hadoop/Spark). Centralized approaches usually require some degree of knowledge of all sensory data, and thus yield detection results with high precision. However, the communication overhead can cost significantly to both the infrastructure and participants' devices. The other type is *distributed* approaches that require to process the data *locally* on each participant's smart device and reach local conclusions/decisions by interacting with neighboring devices through short-range machine-to-machine (M2M) communications such as Bluetooth [8] and WiFi-Direct. For example, computing the maximum value of sensor readings can be done in a distributed manner using the well-known max/min consensus protocol [9]. Distributed approaches scale well with the structural change of the network, but may suffer from a low detection accuracy, if certain detection policy is not carefully designed and enforced. Apart from these two types of approaches, using pattern-matching techniques is another new trend for dynamic event detections [10]. It could be applied both in the CS and distributedly on each participant's smart device.

The temporal property of time-varying events and the large number of participants involved in an event detection process make the application of dynamic event detection quite

Manuscript received November 12, 2015; revised February 15, 2016; accepted February 20, 2016. This work was supported by the National Natural Science Foundation of China under Grant 61300179. Early results were reported at IEEE ICC 2015 [1]. *(Corresponding author: Chi Harold Liu.)*

C. H. Liu is with the School of Software, Beijing Institute of Technology, Beijing 100081, China, and also with the Department of Computer Information and Security, Sejong University, Seoul 143-747, South Korea (e-mail: chiliu@bit.edu.cn).

J. Zhao is with the School of Software, Beijing Institute of Technology, Beijing 100081, China (e-mail: rho.ajax@gmail.com).

H. Zhang is with the Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: honggangzhang@zju.edu.cn).

S. Guo is with the School of Computer Science and Engineering, University of Aizu, Fukushima 965-8580, Japan (e-mail: sguo@u-aizu.ac.jp).

K. K. Leung is with the Electrical and Electronic Engineering and Computing Departments, Imperial College, London SW7 2BT, U.K. (e-mail: kin.leung@imperial.ac.uk).

J. Crowcroft is with the Computer Laboratory, University of Cambridge, Cambridge CB2 1TN, U.K. (e-mail: jon.crowcroft@cl.cam.ac.uk).

Digital Object Identifier 10.1109/JSYST.2016.2533538

a challenging problem. It becomes even harder when energy consumption of smart devices is considered, since participants tend to refuse to cooperate if an application consumes too much energy from their devices. Although existing research efforts declare that power consumption could be reduced by minimizing the active period of time for devices or reducing the packet size [11], [12], so far they have not built up solid mathematical foundation to establish the synergy between energy consumption and detection accuracy. Another key design element is how to motivate participants to contribute high quality data. Along this direction, a large amount of research activities have been performed recently. For example, References [13] and [14] propose mechanisms to motivate participants by either real payment or virtual credit. Nevertheless, regardless of the form of incentives, no platform/CS can afford unlimited budget to recruit participants, and no more sensing tasks can be accommodated when the budget runs out. Finally, over-simplified data model is sometimes used in experiments, such as generating data by normal distribution [15], which is insufficient in many cases. Therefore, real-world-data-based simulations are necessary to verify the performance of proposed approaches.

Toward this end, in this paper, we propose a set of novel dynamic event detection algorithms, to explicitly consider the relationship between energy consumption, detection accuracy, and task budget allocation. The contribution of this paper is summarized as follows.

1) We propose two novel centralized event detection algorithms that make use of the *Minimum Cut* (Min-cut) theory and the support vector machine (SVM) pattern-matching technique.
2) We propose a distributed, energy-efficient event detection framework under task budget constraint, in which participants calculate events' regions by negotiating with their neighbors.
3) We propose a novel utility function from the solution of an optimization problem to embed in the distributed framework, to achieve the long-term detection accuracy while minimizing the energy consumption and providing satisfactory incentives to participants.
4) The effectiveness and flexibility of the proposed algorithms are extensively evaluated by real trace-driven experiments.

This paper is organized as follows. Section II reviews the related research activities. Section III establishes formal model of our system. Section IV describes two centralized event detection approaches, as Min-cut-based and SVM-based algorithms. Section V describes our proposed distributed event detection framework and utility function. Section VI extensively evaluates the performance of the proposed strategy by real trace-driven simulations, and finally Section VII concludes this paper.

II. Related Work

A straight-forward solution for event detection is to predefine some threshold values and trigger alarm messages when sensor readings are higher than the thresholds [12]. This, however, may not fit the spatiotemporal properties of most dynamic events, thus may not be practical. Currently, there are mainly two other categories of approaches to tackle this problem. One of them is the centralized approach. In [7], Sakaki *et al.* take advantage of the real-time nature of Twitter posts to construct an earthquake reporting system. Each Twitter user is regarded as a "sensor." If a user detects some possible events, he/she tweets directly to a CS, where complex models are used to distill earthquake information from these large amounts of tweets precisely.

The other category of approaches is distributed. In [16], Zhu *et al.* propose a completely localized Coordinated wAkeup Scheduling algorithm (CAS), to cooperatively determine sensor wake-up schedules. Without relying on location information, CAS is easy to implement and scalable to network density. Distributed processing/event detection is also an important research direction in control systems. Olfati-Saber presents three novel distributed Kalman filtering algorithms for sensor networks in [17], where each sensor in the network observes part of some process and estimate it based only on local information from its neighbors. Although it is not directly applicable in our application scenario, it is still a promising approach.

Using pattern-matching techniques is another trend for dynamic event detection. It could be applied distributedly on each participant as in [10], or applied in the CS as in [18]. Along this direction, some research works use complicated yet accurate algorithms based on techniques such as SVM [19], Naïve Bayes [20], and the pattern-matching techniques are well summarized and compared in [21]. However, in some cases, the exemplary training data for time-varying events are very hard to generate.

All the above research activities fail to consider the device energy consumption when participating the sensing tasks. Existing research declares that power consumption could be reduced by minimizing the active time of devices [11]. Furthermore, in [12], the authors claim that reducing the packet size can also effectively bring down energy depletion, and [22] proposes an energy-efficient information diffusion protocol for mobile crowd sensing. Reference [23] proposes a system for energy-efficient crowdsourcing of mobile sensor data by exploiting data collection opportunities among mobile phone users such as making phone calls or using applications. However, neither of them has built up solid mathematical foundations to explicitly establish the synergy between energy consumption and detection accuracy (in a distributed manner), as the central theme of our work in this paper.

Another key issue in participatory sensing is how to motivate participants to contribute high quality data. By attracting participants and paying rewards as a return, incentive mechanisms play important roles in guaranteeing a stable scale of participants and improving the accuracy, coverage or timeliness of the sensing results. Following this direction, a large amount research efforts have been conducted recently. Lee and Hoh [13] introduce the reverse-auction based dynamic pricing (RADP) incentive mechanism, where participants send their incentive expectations to the platform, and those with lowest expectations are chosen as winners to perform sensing task. Reference [14] enhances this mechanism by replacing real payment with *virtual credit* to keep a more reasonable price competition. This paper also largely extends our previous work [1], where in this paper 1) a new

centralized event detection algorithm based on SVM pattern-matching technique is proposed and extensively evaluated; 2) computational complexity of min-cut-based centralized event detection algorithm is theoretically given; 3) deep theoretical analysis on the proposed utility function from the solution of an optimization problem is given, including the detailed computational complexity analysis; and 4) all three centralized and distributed are extensively evaluated and compared with existing CAS [16] approach by real trace-driven experiments.

III. System Model

In this paper, we assume that a crowd of smart-device-equipped people move in some urban region, such as park, shopping mall, and plaza. We model the entire sensing region as a two-dimensional (2-D) map, denoted by $\mathcal{M}$. To detect interested events in this region, e.g., noise, people gathering, or some other urban breaking events, the platform publishes related sensing tasks, and those who are willing to take part in the tasks and to contribute data are "participants." Each participant's smart device is embedded with the required sensors for specific sensing tasks. We focus on detecting events by using only one kind of sensor in this paper; however, the proposed framework can be easily applied to other cases where multiple sensors are needed. The sensory data collected by participants are finally used to estimate whether or not some events happen and their whereabouts. We note that detecting strict boundary of event is both a nontrivial and unnecessary task. Instead, we assume that if a mobile node found an event, the target event exists within its sensing range. Therefore, by analyzing the sensory information of participating nodes, we can efficiently get the knowledge of target events.

We assume that each sensing task runs continuously for a long period of time, which contains many consecutive detection cycles. Each detection cycle consists of a *Control* plane and *Data* plane. Necessary control messages exchange occurs in the Control plane, and the Data plane comprises three steps: 1) data collection; 2) transmission; and 3) centralized processing. That is, the participants collect data from the environment, perform some local message exchange and computation if needed, and then transmit the necessary data to a CS for final processing.

We define participants as a set $\mathcal{P} \triangleq \{i | i = 1, 2, \ldots, P\}$, where P denotes the size of the crowd. For the rest of this paper, we refer to a particular participant i and his/her associated collective attributes together as a *participant*, or simply a *user*. These attributes are denoted by a tuple $p_i = (x_i, y_i, v_i, e_i, b_i)$, where x_i and y_i are a participant's coordinates, v_i is the sensory reading collected from his/her smart device, e_i is the device-remaining energy level, and b_i denotes the amount of incentives required by participant i. Also, the initial energy reserve of participant i before performing all sensing tasks is denoted by e_i^0. The incentive can be in the form of real money or virtual credits. Without loss of generality, we simply assume that each sensing task has a fixed incentive budget B. We also assume the participants move around in $\mathcal{M}$ randomly. Mathematically, event detection means to partition all the participating nodes into those nodes in an event (i.e., abnormal nodes), and the other normal nodes. That is, given the information of each user p_i, compute a group of users who are in the event region. Specifically, in participatory sensing campaigns, nodes "in an event region" refer to those with "relatively high" sensory readings. Note that sensor reading can have different meanings depending on applications, e.g., temperature, noise level, and moisture.

In a centralized scenario, we model the sensing region as a graph $G(\mathcal{P}, \mathcal{E})$, where $\mathcal{E} = \{l_{ij} | \forall i \in \mathcal{P}\ \forall j \in \mathcal{P}\}$ is the edge set (l_{ij} of the edge between users i and j). An edge denotes some relationship of two neighboring nodes based on their attributes. Specifically, here the weight w_{ij} for each edge $l_{ij} \in \mathcal{E}$ is calculated as: $w_{ij} = \exp(-|v_i - v_j|/d(i,j))$, where $d(i,j)$ denotes the distance between two participants i and j. This metric reflects the change of sensory readings between two neighboring users.

In a distributed scenario, we assume that a participant has necessary computing capability, and all participants have a common, but tunable wireless communication range denoted by δ. This range can guarantee the successful data transmission and reception higher than an signal-to-noise-plus-interference-ratio (SINR) threshold. Extending this range will enforce devices to use higher transmission power, and since this paper is more application oriented, the related Physical (PHY) layer techniques are out of our scope. We define the *neighbor set* of any participant i as

$$\mathcal{N}(i) = \{j | j \in \mathcal{P}, d(i,j) < \delta\}. \tag{1}$$

For a user i, each of his/her neighbor $j \in \mathcal{N}(i)$ is associated with a utility value U_j (to be explained in detail in Section V) to explicitly represent the benefit of choosing user i. In addition, the *nearest* neighbor of a participant i is defined as the one achieving maximum defined utility value among all one-hop neighbors within the communication range, as

$$\mathcal{N}^1(i) = \arg\max_j U_j \quad \forall j \in \mathcal{N}(i) \tag{2}$$

where U_j is the utility value of user j. Similarly, its *Top-K nearest* neighbors $\mathcal{N}^K(i)$ is a set that contains its K nearest neighbors.

Note that during multiple detection cycles, sensory data from a participant changes accordingly. Thus, all the variables above can be combined with the factor of time. For instance, we use $v_i(t)$ and $e_i(t)$ to denote the time-varying sensory reading and remaining energy of user i in cycle t. However, in centralized algorithms, for the sake of simplicity, we only need to focus on one detection cycle, since all cycles have identical actions (that does not imply the same effect due to system dynamics), and thus we ignore the time factor in Section IV.

Some important notations used in this paper are listed in Table I.

IV. Centralized Approaches

As mentioned above, for centralized algorithms, after participants collect data from the environment, the raw data are transmitted to the CS for further processing. Since data transmission methods (e.g., WiFi, Bluetooth, 3G/LTE) have been thoroughly studied, and the centralized approach does not require control messages exchange before the Data plane starts,

TABLE I
LIST OF IMPORTANT NOTATIONS AND THEIR DESCRIPTIONS

Notation	Explanation
$\mathcal{M}$	Interested region in an application
$\mathcal{P}$	Set of participants
P	Total number of participants
x_i, y_i, v_i, e_i, b_i	The coordinates, sensory reading, remaining energy and incentive requirement of participant i
$d(i,j)$	The Euclidean distance of two participants i and j
w_{ij}	Edge weights between two participant i and j
$\mathcal{E}$	Edges set in graph model $G(\mathcal{P},\mathcal{E})$
$\mathcal{E}_{\mathrm{c}}$	A minimum cut of graph $G(\mathcal{P},\mathcal{E})$
$\mathcal{P}_{in}, \mathcal{P}_{out}$	Subsets of $\mathcal{P}$ that contain the participants that within/out of the events
e_i^0	Initial energy of participant i's device
$\omega_i(k,t)$	Accuracy for participant i to choose user k in cycle t
$A_i(k,t)$	Average of $\omega_i(k,t)$ up to cycle t
$T_i(k)$	Estimated target accuracy for i to choose participant k
$U_i(k,t)$	Utility value of i to choose participant k

Algorithm 1. Min-cut Based Centralized Event Detection Algorithm

1: **Input**: Participants sets $\mathcal{P}$;
2: Initialize graph $G(\mathcal{P},\mathcal{E})$;
3: Removing non-neighboring edges and generate $G'(\mathcal{P},\mathcal{E}')$;
4: $\mathcal{E}_{\mathrm{c}}, \mathcal{P}_{\mathrm{in}} = \emptyset$;
5: $\mathcal{P}_{\mathrm{out}} = \mathcal{P}$;
6: **while** $\max_{\mathcal{P}_{\mathrm{out}}} - \min_{\mathcal{P}_{\mathrm{out}}} > \Delta$ **do**
7: Calculating edge weights;
8: Applying Min-Cut algorithm on G' to identify $\mathcal{E}_{\mathrm{c}}$;
9: Update $\mathcal{P}_{\mathrm{in}}$ and $\mathcal{P}_{\mathrm{out}}$;
10: **end while**
11: Allocate incentives to all participants.
12: **Output**: $\mathcal{P}_{\mathrm{in}}$.

we focus on the data processing step of Data plane in this section. In the following part, we will introduce two novel centralized algorithm, both of which aims at separating nodes in the event area from the other normal nodes.

A. Min-Cut-Based Centralized Algorithm

In $G(\mathcal{P},\mathcal{E})$, the core step is to use an improved Min-cut algorithm [24] to find a subset $\mathcal{E}_{\mathrm{c}} \subset \mathcal{E}$ that connects two neighboring participants *within and out of the area* of an event. Then, all users in an event region can be separated out. The detection algorithm is presented in Algorithm 1.

As an example, Fig. 1 shows the steps of how participants in an event region are recognized in a mini-network with five participants $p_1 - p_5$. Obviously, with its abnormally high reading, user p_5 is identified in an event region. Thus, edges $\overline{p_1p_5}$, $\overline{p_2p_5}$, $\overline{p_3p_5}$, and $\overline{p_4p_5}$ all connect one user inside an event area and one that is outside. Our algorithm can recognize these four edges, as shown in Fig. 1(c). Once these edges are removed, the participants in an event region (just one user p_5 in this simple case) are identified [see Fig. 1(d)]. The detailed process of this algorithm is explained below.

Step 1: Initialization. To initialize $\mathcal{E}$, every two participants in $\mathcal{P}$ are connected, thus $\mathcal{E} = \{l_{ij} | \forall i \in \mathcal{P}\ \forall j \in \mathcal{P}, i \neq j\}$. In other words, a fully connected graph is formed, as shown in Fig. 1(a). The three values besides each node denote the coordinates and sensory reading of each participants, respectively. The information of residual energy and incentive requirement are concealed in this example for simplicity.

Step 2: Edge removal. To reserve only those edges connecting neighboring users, those between nonadjacent users are removed. Note that by "adjacent," we mean users who are within the predefined communication range of each other. Extending this range (e.g., through increasing the transmission power) will allow any particular user to have more neighbors, and the network will become more connected but at the same time more challenging to analyze. Given the coordinates of each edge's vertexes, the number of crosses by other edges can be calculated. To further simplify the fully connected graph, we employ one practical strategy as follows. The edges having most crosses with others are removed from $\mathcal{E}$ iteratively. If two edges have one common intersection, the longer one is removed. Detecting and omitting the crossing links is an essential step in construct a suitable graph for subsequent steps, since a graph with crossing links cannot be processed by Min-cut algorithms properly. After this step, remaining edges consist of a new set $\mathcal{E}'$.

Step 3: Calculating edge weights. The weight w_{ij} for each edge $l_{ij} \in \mathcal{E}'$ is calculated. The bigger sensory readings change among two neighboring vertexes, the lower weight is assigned to the edge l_{ij}, as shown in Fig. 1(b).

Step 4: Min-cut. Finally, the Min-cut algorithm is applied. A *minimum cut* $\mathcal{E}_{\mathrm{c}}$ of a graph is a cut that divide the vertexes into two nonempty disjoint sets $\mathcal{P}_{\mathrm{in}}$ and $\mathcal{P}_{\mathrm{out}}$ and has the smallest sum of weights possible, where $\mathcal{P}_{\mathrm{in}} \cup \mathcal{P}_{\mathrm{out}} = \mathcal{P}$, $\mathcal{P}_{\mathrm{in}} \cap \mathcal{P}_{\mathrm{out}} = \emptyset$, $i \in \mathcal{P}_{\mathrm{in}}, j \in \mathcal{P}_{\mathrm{out}}\ \forall l_{ij} \in \mathcal{E}_{\mathrm{c}}$. In our case, it represents the edges that connect two set that are within and out of the events.

Step 5: Repeat for multiple events. Let $\min_{\mathcal{P}_{\mathrm{out}}} \triangleq \min(v_i)$, $\max_{\mathcal{P}_{\mathrm{out}}} \triangleq \max(v_i)\ \forall i \in \mathcal{P}_{\mathrm{out}}$. Δ is a predefined upper-bound of abnormal reading. Once an event region is detected, the judgment condition $(\max_{\mathcal{P}_{\mathrm{out}}} - \min_{\mathcal{P}_{\mathrm{out}}}) \geq \Delta$ is updated. If there is still another event to be detected, the Min-cut algorithm loops to find other event regions.

Step 6: Incentive allocation. After each detection cycle, the incentives are issued to all participants as they have required, and a new detection cycle resumes from **Step 1** until the budget runs out.

When no more event regions can be detected, edges in $\mathcal{E}_{\mathrm{c}}$ are computed, and the users in $\mathcal{P}_{\mathrm{in}}$ are revealed. These are exactly the participants in the region of events. This algorithm does not assume a predefined threshold for events. Besides, it can detect multiple event regions. Moreover, by traversing all possible edges between any two users, this algorithm provides precise detection result.

However, despite its high precision, it consumes much computational resources and runs slowly, which can be confirmed in the simulation section. According to [24], the Min-cut algorithm has an overall running time of $\Theta(|V||E| + |V|^2 \log |V|)$, where $|V|$ and $|E|$ denote the number of vertexes and edges in a graph, respectively.

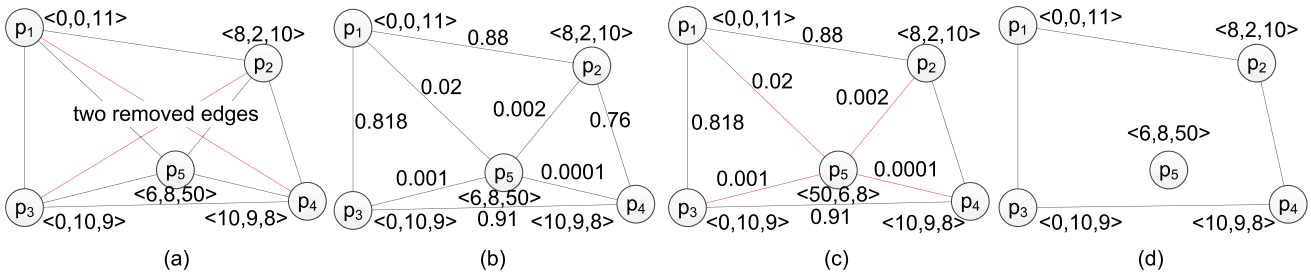

Fig. 1. Example of Min-cut-based centralized algorithm. (a) Connect every two vertexes. (b) Remove edges connecting nonadjacent vertexes and calculate weights. (c) Apply Min-cut algorithm and recognize edges to be removed. (d) Final detection result.

B. SVM-Based Centralized Algorithm

We present a SVM-based centralized algorithm in this section. The intuition is that real-world events usually exhibit some spatiotemporal patterns, and since our data are collected over space and time domains in the sensing region, the data may eventually show these patterns. This intuition can be confirmed by the observations in [25], and then an event detection problem can be transformed into a pattern-matching problem, where SVM technique is used in our approach. SVM is a kind of supervised learning algorithms that used for patterns classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories (in our case, *is* or *is not* in an event region), an SVM training algorithm builds a model that assigns new examples into one category or the other.

Similar with the Min-cut-based algorithm, the participants' information, including their sensory readings and locations, for SVM-based algorithm, are required to be transmitted to the CS. Detailed procedure is explained as follows.

Step 1: Training. To train an SVM, we have to provide a set of exemplary training data, i.e., participants' information and corresponding labels showing whether a user is in an event region or not. The algorithm first acquires a series of labeled training data, which can be collected by running Min-cut-based algorithm. Given training set and proper parameters, we train an SVM model in the CS. Then, the trained SVM model infers an approximate region of the event.

Step 2: Detection. After the training phase is completed, the sensory data are transmitted to the CS for classification. Collected sensory data are fed into the SVM model that will judge whether a user is in an event region or not.

Step 3: Incentive allocation. The SVM-based detection algorithm uses the same budget allocation strategy as the Min-cut-based approach. That is, the required amount of incentives are allocated to all participants at the end of the current detection cycle, and the new detection cycle starts with **Step 1**.

Suppose that the pattern between training data and real sensory data is similar, i.e., the event does not change too much over time and space domains, this algorithm can detect the event region precisely and efficiently.

The time-complexity analysis of running SVM is a non-trivial task, where the costs are coming from the training and test phases. As discussed in [26], the training time depends on both Q^3 (where Q is the number of free support vectors) and nS (where n denotes the number of training samples and S denotes the number of support vectors). In general, although these parameters depend on specific techniques being used, training time is usually expected to be in the order of $O(n^2)$ for state-of-the-art linear SVMs or approximate solvers [26], [27]. On the other hand, in the detection phase, the prediction time is linear with respect to the number of support vectors and features, where [26] discusses it in detail.

Compared to the Min-cut-based algorithm, SVM-based approach runs much faster, and has a good detection precision. However, training sets are required before detection, but the exemplary training data can be very hard to generate in most practical but complicated applications.

C. Summary

Although centralized detection approaches are easy to implement for each participant, it requires all sensory data to be transmitted to the CS for centralized computations. Communication overhead when uploading participants' sensory data can cost significantly to both the infrastructure and participants. Furthermore, they do not scale well with the size and structure change of the network, nor the energy consumption has been clearly considered. To this end, we aim to present a distributed approach in the next section.

V. Distributed Event Detection Approach

Different from all centralized algorithms, in a distributed approach, participants not only collect data, but also need to exchange control messages and make local decisions so that the data can be processed locally. Then they transmit the processed data (e.g., event detection result) back to the CS, where CS has no further processing task to perform. Therefore, the key part of a distributed approach is to design how each user makes decisions locally in an optimal way, as in the Control plane. In this section, we propose a distributed detection framework that provides as accurate detection results as possible, and takes energy conservation into account at the same time. Without loss of generality, we assume that the threshold to recognize an event has already been given to each user, but the value can be easily changed in different applications.

A. Distributed Event Detection Framework

The essence of our proposed approach is described as follows. While each user can detect events according to the

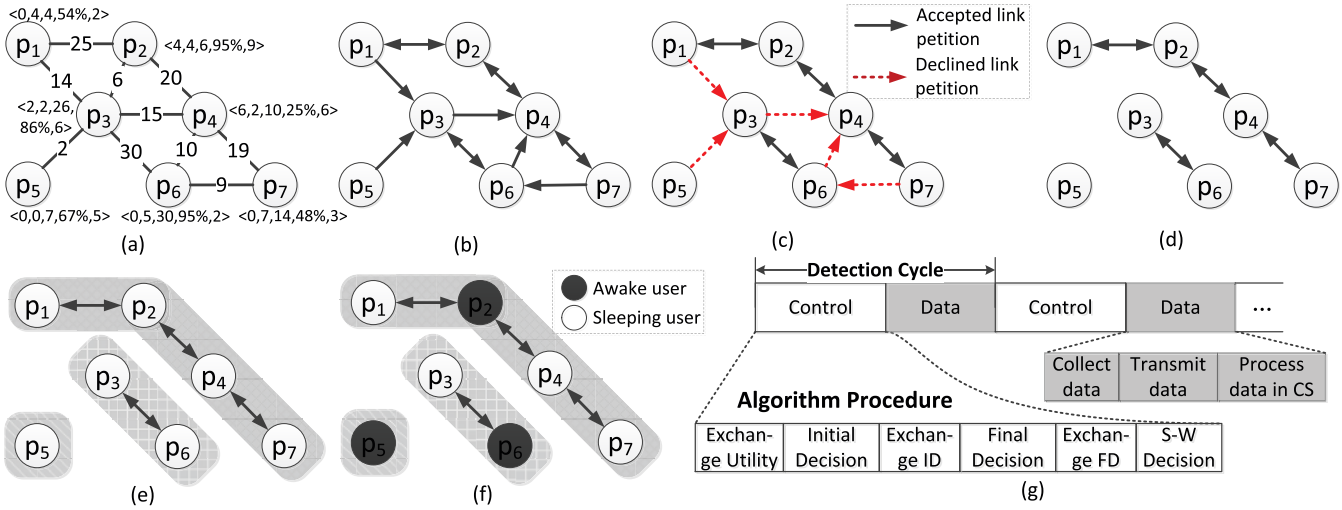

Fig. 2. (a) Information and utility of seven participants. (b) Each participant chooses its Top-2 nearest neighbors as initial decision. (c) By exchanging initial decision information, each participant accepts or declines link petitions. (d) Each participant makes his final decision. (e) By exchanging final decision information, participants form different groups. (f) In each group, a participant keeps awake to perform sensing work, and the others are in sleep state. (g) Employed algorithm procedure of the distributed algorithm.

predefined threshold, many participants tend to provide similar detection results due to their similar geographical locations. Accordingly, we group participants with "similar" neighbors together, so that they are very likely to yield the same detection result. Therefore, in a detection cycle, when one user is monitoring the environment, its *companions*, i.e., other members in the same group, do not have to monitor in the same cycle. As a result, energy is conserved. In this process, a user chooses its companions according to their predefined utility values. A proper utility definition is obviously a vital part in our detection framework, and it will be theoretically discussed in detail later; as for now we focus on the detection framework.

At the beginning of each detection cycle, each user in region $\mathcal{M}$ selects its Top-K nearest neighboring users, referred as the "linking petition" to these neighbors. For convenience, we refer to the initially selected neighbors of participant i as its *desired users*. Conversely, participant i is a *petitioner* of its desired users. Due to the distributed nature of our framework, each user has no *prior* knowledge of its neighbors' initial decisions before making its own decision, and thus it is highly possible that a user makes an improper decision that may create selection conflicts.

Fig. 2(a)–(f) shows an example to illustrate the proposed procedure. Fig. 2(a) shows the initial information of seven participants in a target region. The tuple near each participant represents each his/her coordinates, collected sensory reading, remaining energy level, and incentive requirement. To illustrate a neighboring relationship, all participants are connected to their one-hop neighbors. The number on each line denotes the utility value between two users. In practice, the utility can be derived by using various methods, and it is possible that the utility from p_i to p_j is different from that from p_j to p_i. However, for simplicity, we assume identical utility values in this example. In Fig. 2(b), each user finds its Top-2 (i.e., $K = 2$) nearest neighbors, i.e., the maximum allowed number of companions are two in this example. An arrow in the figure points from a petitioner to its desired user. However, we note that, in our case, user p_5 chooses p_3 whereas p_3 chooses his/her another two neighbors p_4 and p_6, which creates a conflict situation. Our solutions is that: if participant p_i finds its desired user p_j's desired user is not p_i itself, p_i gives up its petition. The reason is that from p_j's perspective, by connecting its own desired user instead of p_i, it could achieve better accuracy. Therefore, in Fig. 2(c), the linking petition of user p_5, together with other unqualified petitions, is declined after their initial decisions are exchanged. The result of their petition decisions is shown in Fig. 2(d). Fig. 2(e) and (f) shows that users exchange information and decide to stay awake or sleep during this detection cycle. Note that although not mentioned, we assume that the communication range can assure one-hop communication between users. Below is a formal description of our distributed event detection framework.

Step 1: Exchange user information. Each user sends its equipped sensor information to its neighbors. Based on this, each user computes the utility of its neighbors.

Step 2: Make initial decision. According to the associated utility, each user chooses K neighbors to be its desired users.

Step 3: Exchange initial decision. Each user's initial decision is transmitted to all neighboring participants by sending control messages.

Step 4: Make petition decision. With knowledge of each user's petitioner, the strategy described above is applied to decide whether it should accept linking petitions from its petitioners.

Step 5: Exchange petition decision. Each user notifies its petitioners whether they are accepted. Then, each user makes its final decision on whether it should form a group with its desired users or not. Finally, each user either forms a group with other users or stands alone.

Step 6: Make sleep–wake (S–W) decision. Once a group is formed, a node exchanges the information within its own group.

Each user decides whether herself should maintain sleeping state in this cycle. If she is not in a group, she stays awake, otherwise she will check its companions' residual energy and whoever has more energy stays awake during this cycle.

Step 7: Incentive allocation. Awake users are issued their required amount of incentives, and a new detection cycle starts with **Step 1**.

After running the above algorithm distributedly, participants start to collect data from the environment, as in the Data plane. If any event is detected, the corresponding users send alerts to the CS that contains information as short as 1 bit to notify the CS that an event happens within its detection range.

B. Computational Complexity Analysis

Our proposed framework requires exchanges of three types of control messages: the utility value of each neighbors, the initial decision message (ID), and final decision message (FD) of each participant. Besides, the participants also need to make local decisions on his/her ID, FD, and S–W status. The distributed algorithm procedure and its relationship with the whole detection cycle is presented in Fig. 2(g). The Control plane contains all six steps of our proposed framework. Thus the runtime of the framework consists of the time periods to complete: 1) the control message exchange as in Steps 1, 3, and 5; and 2) the decision-making process as in Steps 2, 4, and 6. In the information exchange steps, each participant only needs to broadcast his information to his neighbors. We could reasonably assume that these steps all run within a fixed amount of time. Therefore, the running time mainly depends on Steps 2, 4, and 6.

Consider a network in which average neighbor number of each user is μ. In the graph $G(\mathcal{P}, \mathcal{E})$ where users in $\mathcal{P}$ are all connected to their one-hop neighbors by edges in $\mathcal{E}$, $\mu = 2N_{\text{edge}}/N_{\text{vertex}}$ (N_{edge} is the number of edges and N_{vertex} is the number of users). Besides, we assume that all information exchange steps last for T_1 units of time, and all the primitive operations take a fixed amount of T_2 time units. Thus, running time analysis is performed as follows.

1) All three information exchange steps take T_1 time units.
2) In Step 2, for each user, utility information of its μ neighbors are stored in its local memory. A simple *max* procedure is applied to find the neighbor with maximum utility. A analysis of this for-loop procedure is given below.
   In worse case where the statements in the for-loop have to be run in every iteration, the running time for this phase is $\mu \times 6T_2 + 2T_2$.
3) In Step 4, the main part is also a for-loop. Similarly, analysis goes here. A loop contains computation of one absolute operation, one minus operation ($2T_2$), three value fetches ($3T_2$), a comparison (T_2), label a petitioner whether it is acceptable or not (one value fetch and one store = $2T_2$). Thus, the running time for this phase is approximately $\mu \times (5T_2 + T_2 + 2T_2)$.
4) In Step 6, each user fetches its own energy value and its companion's, then compare them (two value fetches + one operation = $3T_2$).

The total running time for our proposed distributed algorithm is equal to

$$3T_1 + \mu \times 6T_2 + 2T_2 + \mu \times 8T_2 + 3T_2 \approx 3T_1 + 14\,\mu T_2. \tag{3}$$

Since T_1, T_2, and μ are all constants, the running time of this algorithm is in the order $\Theta(1)$. In other words, the distributed approach is a scalable approach whose running time is independent of participants numbers.

To summarize, the distributed approach has the following advantages.

1) It runs fully distributedly.
2) It achieves linear time complexity, and can be significantly faster than the centralized algorithms.
3) It is robust and scales well with the size or structural change of the network.
4) It is energy-efficient. By sending alert information when necessary instead of raw data, it reduces the energy consumption. Furthermore, by S–W scheduling in the detection process, the energy is reserved to a large extent.

C. Energy-Efficient Utility Definition

To complete our proposed distributed event detection algorithm, a proper definition of the utility is important. Theoretically any proper function could be plugged in our framework to compute a utility of a user. For example, a simple random-neighbor strategy could be applied, where a participant randomly chooses a neighbor to make the initial decision. However, we need to build up explicit synergies between detection accuracy, energy consumption, and incentive allocation for all user.

We assume that each participant has an average sensory reading $\bar{v}_i(t)$ up to cycle t. An intuition is to let a user pick the neighbor that has the most similar average reading with itself. In this regard, we calculate the *target accuracy* of neighbor k of any user i as

$$\boldsymbol{T_i}(k,t) = \exp\left(-\left|\bar{v}_i(t) - \bar{v}_k(t)\right|\right). \tag{4}$$

However, when a participant's device energy consumes, the detection accuracy is also decreased. Thus, a neighbor with more remaining energy and less incentive requirement should be given higher priority to join the detection task. Therefore, for user i, we define its possible *achieved accuracy* when selecting neighbor k to complete a detection task at cycle t, as

$$\omega_i(k,t) = \exp\left(-\left|v_i(t) - v_k(t)\right|\right) \times \frac{\phi_k(t)}{\phi_0} \tag{5}$$

where

$$\phi_k(t) = \frac{e_k(t)}{b_k(t)}, \quad \phi_0 = \frac{e_k^0}{b_{\max}} \tag{6}$$

$b_{\max}$ denotes the highest required incentive among all participants. $\phi_k(t)$ combines both the energy consumption and incentive allocation factors, representing the affordable remaining energy per unit budget costs of neighbor k at detection cycle t, and then ϕ_0 normalizes $\phi_k(t)$.

Suppose that user i has m neighbors, then our goal is to define an appropriate utility with which the *long-term average accuracy* of all neighbors $\boldsymbol{A_i} = \{A_i(1), A_i(2), \ldots, A_i(m)\}$ is proportional to the target accuracy $\boldsymbol{T_i}$. Here, $A_i(k,t) = \sum_{l=1}^{t} w_i(k,l)/t$. In order to define the utility, we decompose the analysis process into two lemmas. The first one formulate an optimization problem for any user i, then the second lemma proves that the optimal solution A_i^* to the optimization problem is proportional to the target $\boldsymbol{T_i} = \{T_i(1), T_i(2), \ldots, T_i(m)\}$. Note that for simplicity, we drop the time factor in these lemmas.

Lemma V.1: If each user maximizes the following objective function over A_i:

$$\begin{aligned} \max f(A_i) &= \sum_{k=1}^{m} T_i(k) \cdot \log\left(A_i(k)\right) \\ \text{s.t.} \sum_{\mathrm{k=1}}^{\mathrm{m}} \mathrm{A_i(k)} &\le m \\ \sum_{k=1}^{m} b_i(k) &\le B \end{aligned} \tag{7}$$

where m upper bounds the total average accuracy of all neighbors of user i. The upper bound is achieved when $\omega_i(k,t) = 1\ \forall k = 1, 2, \ldots, m$. Then, the optimal solution A_i^* is proportional to T_i.

Proof: This problem is a classic constrained optimization problem, thus could be solved with Lagrange multipliers. Specifically, we have

$$\begin{aligned} L = &\sum_{k=1}^{m} T_i(k) \cdot \log\left(A_i(k)\right) - \lambda_i^1 \cdot \left(\sum_{k=1}^{m} A_i(k) - m\right) \\ &- \lambda_i^2 \cdot \left(\sum_{k=1}^{m} b_i(k) - B\right). \end{aligned} \tag{8}$$

The first-order (necessary) optimality condition for (8) is

$$\bigtriangledown L = 0 \ \text{ and } \ \lambda_i \left(\sum_{k=1}^{m} A_i(k) - m\right) = 0. \tag{9}$$

Since the constraint is binding and $\lambda \neq 0$, the first part in (9) could be solved as

$$\frac{T_i(k)}{A_i^*(k)} = \lambda_i. \tag{10}$$

This means that after some iterations the average accuracy $\boldsymbol{A_i^*}$ is proportional to $\boldsymbol{T_i}$ element-wisely. ■

Then, we define a utility function and prove that if this utility is enforced in our proposed distributed framework, the objective function in (7) converges to the optimal solution.

Lemma V.2: If user i uses the following utility for its neighbors:

$$U_i(k,t) = T_i(k)\frac{\omega_i(k)}{A_i(k)} \tag{11}$$

the algorithm maximizes the objective function in (7) iteratively.

Proof: Since the objective function in (7) is convex, the sufficient and necessary condition of optimality for this problem is

$$\bigtriangledown f|_{\boldsymbol{A_i}} \cdot (\boldsymbol{A_i} - \boldsymbol{A_i^*}) \le 0 \tag{12}$$

where $\boldsymbol{A_i}$ could be any arbitrary energy consumption vector. This equation could be further broken into two parts

$$\sum_{k=1}^{m} T_i(k)\frac{A_i(k) - A_i^*(k)}{A_i(k)} \tag{13}$$

where $A_i(k)$ and $A_i^*(k)$ are the average of $e_i(k)$ in time, so this equation could be rewrite as

$$\sum_{k=1}^{m} T_i(k)\frac{\mathbb{E}[\omega_i(k)]}{A_i(k)} - \sum_{k=1}^{m} T_i(k)\frac{\mathbb{E}[\omega_i^*(k)]}{A_i(k)} \tag{14}$$

where $\mathbb{E}$ denotes the expectation of target accuracy for a series of cycles. Then, maximizing the following will maximize the second part of (14):

$$\max_{\omega_i} \quad T_i(k)\frac{\omega_i(k)}{A_i(k)}. \tag{15}$$

Since e_i in the first part in (14) is not optimal as in (15), the second part must be greater than the first term in (14). So this equation holds. That is, use the metric in (11) maximizes the objective function in (7). ■

In practice, let us put the time factor back into consideration, and (11) is rewritten as

$$U_i(k,t) = T_i(k)\frac{\omega_i(k,t-1)}{A_i(k,t-1)}. \tag{16}$$

VI. Performance Evaluation

In this section, we first present the simulation settings and discuss the used real data set, and then show the simulation results.

A. Simulation Settings

We assess them with the Microsoft Research Asia GeoLife data set [28], where real movement traces of ordinary citizens are used to represent mobile users in the considered scenario. The GeoLife project has collected 182 volunteers' trajectories in an urban area of Beijing for 3 consecutive years.

As all traces spread in different parts of Beijing, a specific rectangular region where the traces mostly appear is needed. We thus store all trajectories in a geographical MySQL database and find a $200 \times 500\ \mathrm{m}^2$ region that is of high user density, as shown in Fig. 3(a), which happens to be around the area of the Microsoft Research Asia site. There are totally 612 trajectories in the considered region. Each trajectory is marked by a sequence of time-stamped GPS points that contain users' latitude, longitude at a given time. Fig. 3(b) shows all 612 user trajectories. Since these trajectories are recorded at different

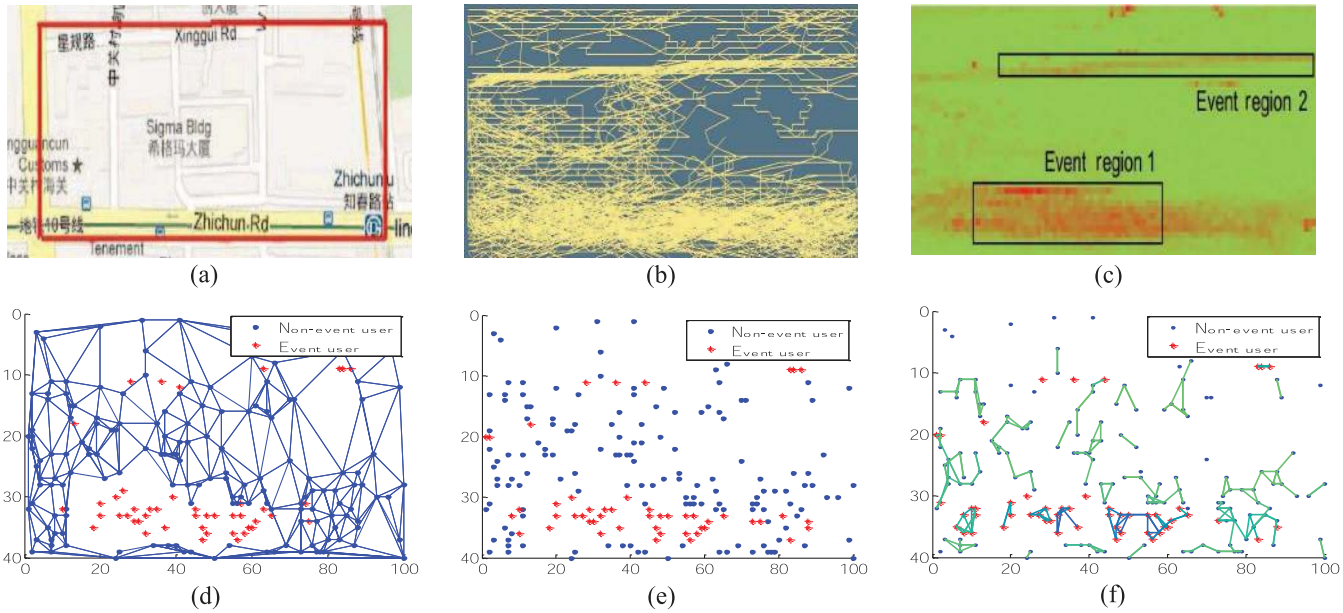

Fig. 3. (a) Simulation region in Beijing, China (red rectangle). (b) User trajectories. (c) Generated noise levels based on overlaid user trajectories. (d)–(f) Detection results of Min-cut-based, SVM-based, and distributed algorithms, respectively.

times, in our simulations, we simply neglect their time index and overlay them into the same time period.

Based on this data set, we consider a noise level monitoring application in this area. Without loss of generality, we use the number of visits on each area for a certain period of time to indicate the noise level (i.e., more visit can potentially indicate that that area is more crowded and thus with higher noise level, compared with sparsely visited areas). Specifically, we consider the noise level data as the sensory reading. Given that the best GPS accuracy is about 5 m, we then divide the whole region into 40×100 small cells, each with a size of 5 m$\times$5 m. We use them as the participants' location instead of the original GPS coordinates, i.e., the horizontal axis of the whole map $\mathcal{M}$ ranges from 0 to 40 and vertical axis is from 0 to 100. Fig. 3(c) shows the generated noise levels, where we observe two concentrated noisy regions.

The communication range of each participant is set to 10 m. We employ the energy dissipation model [29], and the energy cost to transmit a L-bit message between two users is given by

$$E_{\rm tx}(L,d) = \begin{cases} L\varepsilon_{\rm tx} + L\varepsilon_{\rm fs} d^2, & \text{if } d < d_0 \\ L\varepsilon_{\rm tx} + L\varepsilon_{\rm mp} d^4, & \text{if } d \geq d_0 \end{cases}$$

where d denotes the distance between two users, $\varepsilon_{\rm tx}$ is the energy dissipation per bit to run the transmitter circuit, $\varepsilon_{\rm fs}$ and $\varepsilon_{\rm mp}$ are transmitter amplifiers, and $d_0 = \sqrt{\varepsilon_{\rm fs}/\varepsilon_{\rm mp}}$. We adopt the same parameter settings in [29]: $\varepsilon_{\rm tx} = 50\,{\rm nJ/bit}$, $\varepsilon_{\rm fs} = 10\,{\rm pJ/bit/m^2}$, and $\varepsilon_{\rm fs} = 0.0013\,{\rm pJ/bit/m^4}$. The energy dissipated in event monitoring per round is 0.001 mJ, and the initial energy of each user is set to 5 mJ. There are three different kinds of packets transmitted in the network: the raw data packet in centralized algorithms, the choice packet exchanged between neighbors in the proposed distributed algorithm, and the event alert packet a user sends to the CS. Their sizes are set to 256, 128, and 64 bits, respectively.

B. Results and Discussion

We assess the performance of our proposed Min-cut/SVM-based centralized algorithms as the benchmark for our proposed distributed approach, where the defined utility and simply selecting a random neighbor are also compared. Besides, to validate the performance of our proposed distributed algorithm, we also compare it with "CAS" approach [16], i.e., an energy-efficient and totally localized algorithm for event detection. The key idea of CAS is to evenly distribute the wakeups of each node and its neighbors.

First, we show the detection results of the three proposed algorithms. We use 200 participants in the simulations. As shown in Fig. 3(d)–(f), all three approaches successfully divide users into two groups: users that are inside, and out of the event region, but the results are presented in different manners. That is, Min-cut-based centralized algorithm connect all non-event users together, leaving out those with high noise levels in the event region [see Fig. 3(d)]; SVM-based algorithm directly classifies whether each user is in an event region with trained model [see Fig. 3(e)]; and users form local groups by our proposed distributed algorithm [see Fig. 3(f)], where members in each group share common detection results. Lines with deeper color represent higher noise levels, as the event region.

Next, we verify the convergence of our proposed utility in the distributed detection framework. We set a simple scenario where one user i interacts with six neighbors. In each iteration, participant i selects one of his/her neighbors according to their utilities specified in (16). We assign each user a random sensory reading, to compute the target accuracy $T_i(k)\ \forall k = 1, 2, \ldots, 6$. The time-averaged accuracy of each edge is also recorded and compared to the target accuracy. To validate the convergence performance, we propose to use a ratio $\alpha_i(k,t) = A_i(k,t)/T_i(k)$. Since our target in this utility is to let $A_i(k,t)$ and $T_i(k)$ be proportional to each other, it would be ideal if

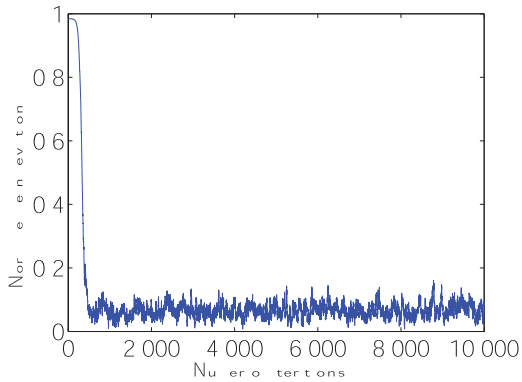

Fig. 4. Convergence performance of the proposed utility.

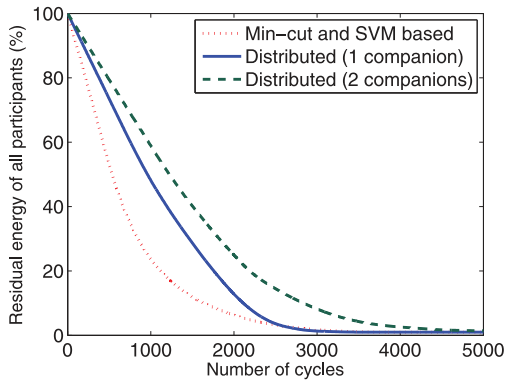

Fig. 5. Network lifetime performance: remaining energy of all participants after each detection cycles of different approaches.

TABLE II
RUNNING TIME OF THREE ALGORITHMS WITH DIFFERENT NUMBER OF PARTICIPANTS (STANDARD DEVIATION)

No. of users	100	200	300	400	500
Min-cut (min)	0.1 (±0.02)	1.33 (±0.06)	5.97 (±0.28)	14.77 (±0.35)	57.22 (±1.94)
SVM (s)	0.0435 (±0.0014)	0.0441 (±0.0019)	0.0447 (±0.0024)	0.0458 (±0.0035)	0.0472 (±0.0147)
Distributed (s)	0.0045 (±0.0001)	0.0095 (±0.0001)	0.0143 (±0.0001)	0.0191 (±0.0001)	0.0239 (±0.0001)

$\alpha_i(k,t)$ for all k becomes similar given a large enough t. We use the normalized mean deviation of α for each iteration, as: $C_i(t) = \frac{1}{6}\sum_{i=1}^{6} |(\alpha_i(k,t) - \mu(t))/\mu(t)|$, where $\mu(t)$ is the mean of $\alpha_i(k,t)$. Fig. 4 shows that our proposed algorithm successfully converges to the target accuracy after around 400 iteration steps.

We then compare the running time of our proposed algorithms in each detection cycle with different number of participants in the sensing region. As shown in Table II, the Min-cut-based algorithm runs in the scale of minutes, and sometimes even hours, which is unacceptable in most applications, let along real-time multimedia services. On the contrary, our proposed distributed approach can run 9.67 times faster than the SVM-based algorithm. It can also be observed that with the state-of-the-art solvers and optimized codes, the running time of SVM does not change dramatically with the increase in the number of participants in this order of magnitude.

Fig. 5 shows the total energy consumption with 200 users. When a user has less than 1% energy left, it is regarded as a dead user. When the detection process continues, the distributed algorithm consumes much less energy than two centralized algorithms. When the maximum allowed companion number increases, the energy consumption decreases further. Fig. 6(c) shows the network lifetime. With the increase in the number of participants, the network lifetime achieved by Min-cut-based centralized detection algorithm barely grows, while our proposed distributed algorithm achieves much higher and increases significantly. Here we define a network (or sensing region) as "not functional," when 90% users run out of battery.

We also observe that the network lifetime is prolonged 38.3% achieved by our proposed algorithm, than that of the Min-cut-based centralized approach, and 86% more if the companion number increases from 1 to 2. CAS's network lifetime performance is better than the centralized algorithm. However, our distributed algorithm can still (with one companion) outperforms it by 13%, because our solution not only consider a user's neighbors but also all users who have similar situation around, by link petition mechanism.

Since the centralized algorithms detect events more accurately, we use its detection result as the performance benchmark. As for the distributed algorithm, we investigate its performance with two utilities: our proposed one in (16) and the random-neighbor strategy (i.e., to randomly select a neighbor to make initial decision). We use detection *precision*, *recall* as two metrics to evaluate their performance, calculated as follows. Suppose TP denotes the number of detected event users that are true event users, FP denotes the number of detected event users that are actually nonevent users, and FN denotes the number of detected nonevent users that are actually event users, the precision (P) and recall (R) are then defined as $P = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$. $R = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$. Fig. 6(a) and (b) shows the results. It can be seen that our proposed solution can maximally achieve 86% precision and 82% recall, if compared with the optimal Min-cut-based algorithm. Besides, our proposal shows better performance than the SVM-based centralized approach. That is, on average, it achieves 4.3% more precision and 9.5% more recall. Since the focus of CAS approach is on the coverage reliability rather than the detection accuracy, it performs better than the random-neighbor approach, but are outperformed by all other methods.

Next, we explore the impact of using different maximum allowed number of companions on the performance of our distributed algorithm. Specifically, each user can choose its Top-K nearest neighbors to form a group, where K ranges from 1 to 4. Fig. 6(d) and (e) shows the detection precision performance with different number of participants. Similarly, Fig. 6(f) shows the network lifetime. It is clear that the more neighbors one user can choose and the more enduring a network is, the less precise the detection result becomes. Despite the number of participants, if the companions number is set as 4 instead of 1, the network lifetime is prolonged 85% on average. On the other hand, the detection precision and recall also decrease 13.1% and 25.6%, respectively.

Then, we investigate the impact of task budget. We compare our proposed distributed approach with two centralized algorithms, which serve as the performance benchmark. In

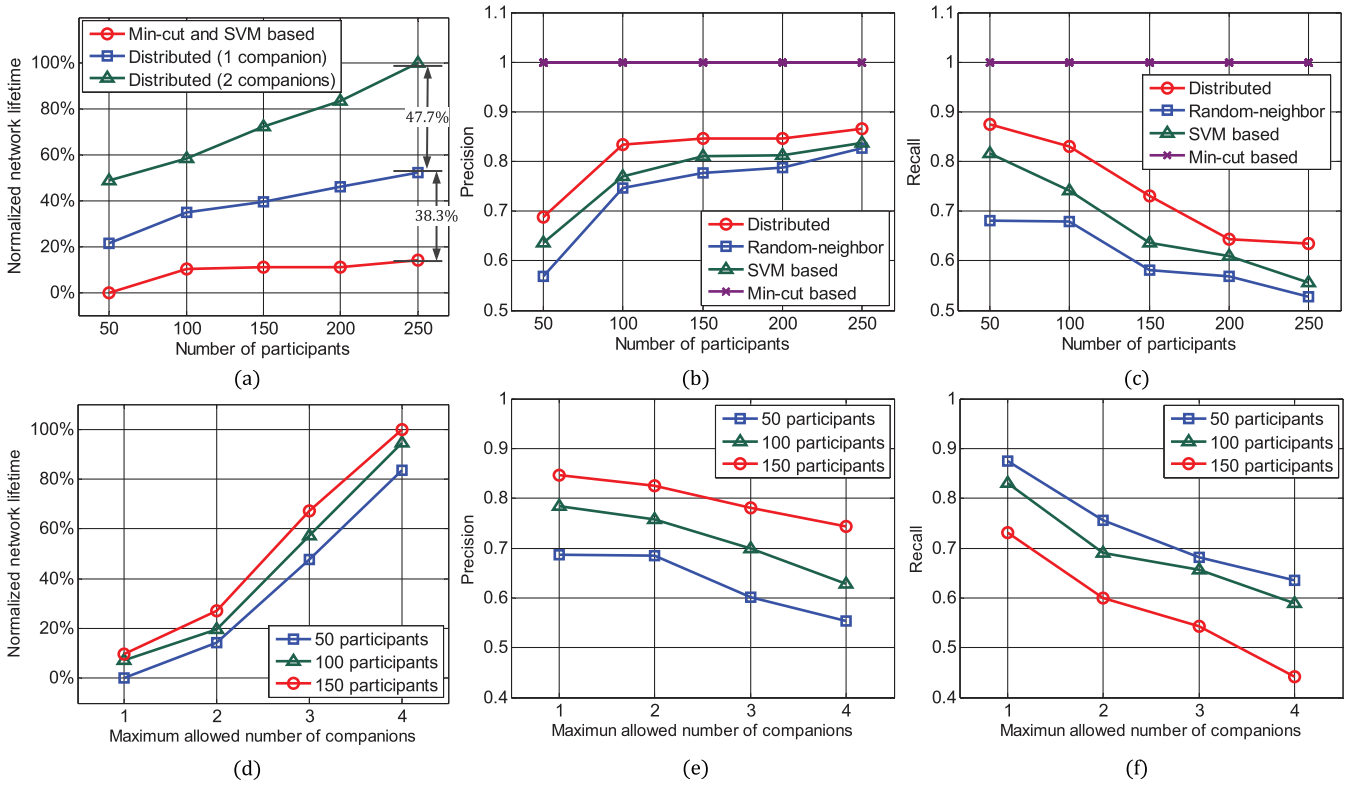

Fig. 6. (a)–(c) Precision, recall, and network lifetime with different number of participants. (d)–(f) Precision, recall, and network lifetime performance with different maximum allowed number of connected neighbors.

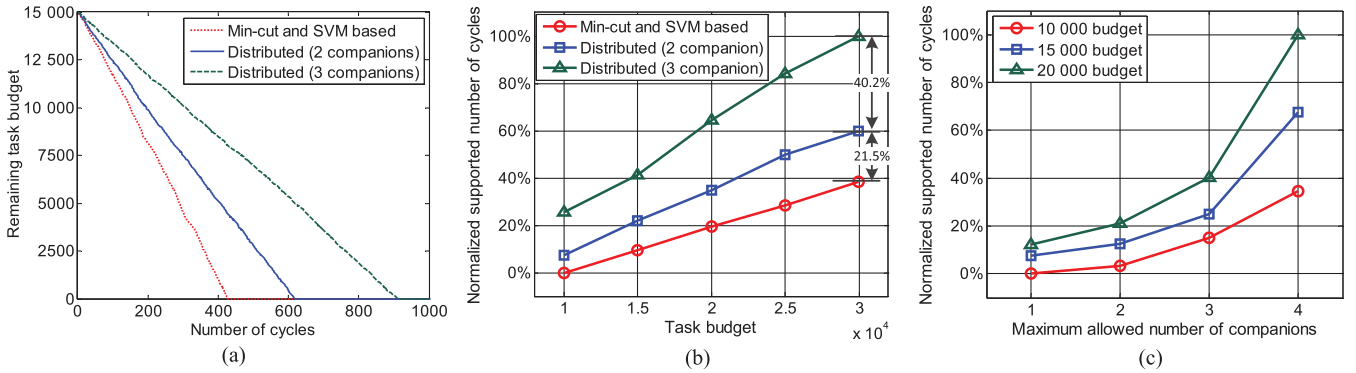

Fig. 7. Simulation results of experiments on budget factor. (a) Remaining incentive budget after each detection cycle. (b) Supported number of detection cycles versus task budget. (c) Supported number of detection cycles versus maximum allowed companion number.

our proposed distributed algorithm, we change the maximum allowed number of companions, and after the task budget runs out, the sensing task will no longer be supported. To better examine the effect of budget constraint, we suppose that participants' smart devices have ample initial energy reserve during the lifetime of sensing tasks. Fig. 7(a) shows the trend of remaining task budget after participants are selected and paid at each detection cycle. We observe that two centralized algorithms spend much faster than the distributed approach, and tasks are no longer supported after around 400 tasks. In comparison, distributed algorithm endures much longer, as 600 tasks, and can be even longer (i.e., 900 tasks) when the allowed number of companions are 2 and 3, respectively.

We next investigate the impact of task's total incentive budget on different approaches. For three approaches, we increase the task budget from 10 000 to 30 000 units. Fig. 7(b) shows that the supported number of detection cycles by two centralized algorithms are 21.5% lower than our proposed distributed approach. When the companion numbers increases from 2 to 3, the supported number of cycles by distributed algorithm also increases to 61.7% higher than the centralized algorithms. This is due to the fact that more budget leads to more supported cycles. Moreover, in our distributed algorithm, when a participant is in sleeping state, he/she does not have to be granted a share of budget, whereas the centralized algorithms recruit all participants during a sensing task.

In the proposed distributed algorithm, we explore the impact of using different companions on its supported number of detection cycles. We allow the maximum allowed companion number K to range from 1 to 4. Fig. 7(c) shows the achieved task duration with three different amount of budget: 10 000, 15 000, and 20 000 units. From this figure, we observe that with more task budget and/or by allowing more companions, the task endures longer. If K is set to 4 instead of 1, the task duration can be prolonged 60.77% on average. Note that the benefit of supported cycles per unit companion also increases when K becomes bigger. That is, when K increases from 1 to 2, the task duration is prolonged 5.75% on average, and this number goes to 14.46% and 40.56% when K changes from 2 to 3 and 3 to 4, respectively. This is because that our distributed framework allows for more companions, and thus potentially more sleeping participants in the sensing task, consequently saving more budget expenses.

VII. Conclusion and Future Work

Dynamic event detection like noise level and air pollution by using participatory sensing paradigms is a promising research direction. In this paper, we first proposed two novel centralized detection algorithms, based on Min-cut theory and SVM pattern recognition techniques as the performance benchmark. To solve the computational complexity problem of centralized approaches, we proposed a distributed detection framework, where an optimization problem is formulated to derive an optimal utility that ensures the long-term detection precision and energy-efficiency of the algorithm under task budget constraints. Extensive experimental results, based on a real trace-driven data set in an urban area of Beijing, showed that our proposed distributed algorithm successfully detect events efficiently, accurately in an energy-efficient manner while minimizing the allocated task budget to all participants. As for the future, we plan to investigate the impact of participant contact frequencies on the system performance.

References

[1] J. Zhao, C. H. Liu, M. Chen, X. Lu, and K. K. Leung, "Energy-efficient dynamic event detection by participatory sensing," in *Proc. IEEE Int. Conf. Commun. (ICC'15)*, 2015, pp. 3180–3185.

[2] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: Scalable sound sensing for people-centric sensing applications on mobile phones," in *Proc. ACM 7th Int. Conf. Mobile Syst. Appl. Serv. (MobiSys'09)*, 2009, pp. 165–178.

[3] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, and B. Nath, "Real-time air quality monitoring through mobile sensing in metropolitan areas," in *Proc. 2nd ACM SIGKDD Int. Workshop Urban Comput. (UrbComp'13)*, 2013 p. 15.

[4] S. Mathur *et al.*, "Parknet: Drive-by sensing of road-side parking statistics," in *Proc. ACM 8th Int. Conf. Mobile Syst. Appl. Serv. (MobiSys'10)*, 2010, pp. 123–136.

[5] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: An end-to-end participatory urban noise mapping system," in *Proc. ACM/IEEE 9th Int. Conf. Inf. Process. Sensor Netw. (IPSN'10)*, 2010, pp. 105–116.

[6] C. Vu, R. Beyah, and Y. Li, "Composite event detection in wireless sensor networks," in *Proc. IEEE Int. Perform. Comput. Commun. Conf. (IPCCC'07)*, Apr. 2007, pp. 264–271.

[7] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: Real-time event detection by social sensors," in *Proc. ACM 19th Int. Conf. World Wide Web (WWW'10)*, 2010, pp. 851–860.

[8] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller, "A system for distributed event detection in wireless sensor networks," in *Proc. ACM/IEEE 9th Int. Conf. Inf. Process. Sensor Netw. (IPSN'10)*, 2010, pp. 94–104.

[9] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[10] E. Ould-Ahmed-Vall, B. H. Ferri, and G. F. Riley, "Distributed fault-tolerance for event detection using heterogeneous wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 12, pp. 1994–2007, Dec. 2012.

[11] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 537–568, 2009.

[12] Y. Li, C. Ai, C. T. Vu, Y. Pan, and R. Beyah, "Delay-bounded and energy-efficient composite event monitoring in heterogeneous wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 9, pp. 1373–1385, Sep. 2010.

[13] J.-S. Lee and B. Hoh, "Sell your experiences: A market mechanism based incentive for participatory sensing," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom'10)*, 2010, pp. 60–68.

[14] J.-S. Lee and B. Hoh, "Dynamic pricing incentive for participatory sensing," *Pervasive Mobile Comput.*, vol. 6, no. 6, pp. 693–708, 2010.

[15] Z. Zhou and G. Qu, "An energy efficient adaptive event detection scheme for wireless sensor network," in *Proc. IEEE Int. Appl.–Specific Syst. Archit. Processors (ASAP'11)*, Sep. 2011, pp. 235–238.

[16] Y. Zhu, Y. Liu, L. Ni, and Z. Zhang, "Low-power distributed event detection in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (INFOCOM'07)*, May 2007, pp. 2401–2405.

[17] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. IEEE 46th Conf. Decis. Control*, Dec. 2007, pp. 5492–5498.

[18] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *Proc. ACM Int. Conf. Manage. Data (SIGMOD'06)*, 2006, pp. 145–156.

[19] V. Traag, A. Browet, F. Calabrese, and F. Morlot, "Social event detection in massive mobile phone data using probabilistic location inference," in *Proc. IEEE 3rd Int. Conf. Soc. Comput. (SocialCom'11); Privacy, Secur., Risk Trust (PASSAT'11)*, Oct. 2011, pp. 625–628.

[20] K. Tang, L. Fei-Fei, and D. Koller, "Learning latent temporal structure for complex event detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR'12)*, Jun. 2012, pp. 1250–1257.

[21] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Commun. Surv. Tuts.*, vol. 12, no. 2, pp. 159–170, May 2010.

[22] T. Higuchi, H. Yamaguchi, T. Higashino, and M. Takai, "A neighbor collaboration mechanism for mobile crowd sensing in opportunistic networks," in *Proc. IEEE Int. Conf. Commun. (ICC'14)*, Jun. 2014, pp. 42–47.

[23] N. D. Lane *et al.*, "Piggyback crowdsensing (PCS): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proc. ACM 11th Conf. Embedded Netw. Sensor Syst. (SenSys'13)*, 2013, pp. 7:1–7:14.

[24] M. Stoer and F. Wagner, "A simple min-cut algorithm," *J. ACM*, vol. 44, no. 4, pp. 585–591, 1997.

[25] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Wireless Sensor Networks*. New York, NY, USA: Springer, 2004, pp. 307–322.

[26] L. Bottou, C.-J. Lin, "Support vector machine solvers," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. Decoste, J. Weston, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 301–320.

[27] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, 2005.

[28] Y. Zheng, X. Xie, and W. Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–40, 2010.

[29] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.