Energy-Efficient Dynamic Event Detection by Participatory Sensing

Jianxin Zhao[†], Chi Harold Liu[†], Min Chen[♮], Xue Liu[‡] and Kin K. Leung[♭]
[†]Beijing Institute of Technology, China, [‡]McGill University, Canada
[♮]Huazhong University of Science and Technology, China, [♭]Imperial College, UK
E-mail: [†]{2220130566,chiliu}@bit.edu.cn, [‡]xueliu@cs.mcgill.ca, [♮]minchen2012@hust.edu.cn, [♭]kin.leung@imperial.ac.uk

***Abstract*—Dynamic event detection by using participatory sensing paradigms has received growing interests in recent years, where detection tasks are assigned to smart device users who can potentially collect needed sensory data from the equipped sensors. These data can be utilized to detect interested events like noise, air pollution, or even earthquake. Since most existing solutions focus on centralized detection approaches that, however, usually cause heavy communication overhead, it is strongly desired to design distributed solutions to reduce energy consumption while achieving a high level of detection accuracy. In this paper, we first present a novel *Minimum Cut* based centralized detection algorithm as the performance benchmark, and then introduce a novel distributed, energy-efficient solution, where an optimization problem is formulated and an optimal solution is derived. Simulations based on a real-trace driven data set in Beijing demonstrate the effectiveness of our proposed algorithms.**

I. Introduction

Participatory sensing is an emerging paradigm that aims at collecting data from a huge amount of smart devices carried by users. These devices are equipped with a variety of embedded sensors, e.g., accelerometer, GPS, light sensor, etc. Participatory sensing can successfully reduce the deployment costs, and enables the sensing of the world at an unprecedented spatio-temporal granularity. These features make it quite suitable to be applied in a wide range of applications; for example, noise [1] and air quality monitoring [2] in urban areas, and urban street-parking availability monitoring [3], etc.

Among many applications of participatory sensing, dynamic event detection is an important one that has received a growing amount of research attention. Our research is motivated by the application scenario in Fig. 1, where a group of smart device users inside a certain spatial sensing region subscribes 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 the necessary data to the CS via built-in communication interfaces, such as 3G/LTE or WiFi. With these data, CS is able to identify the area of target events inside the sensing region with certain accuracy. Such events could be abnormal noise in a specific region, residential fire in forests, or meteorological hazards [4], [5], etc. The temporal property of events and the large amount of participants involved make dynamic event detection a challenging problem; and can be more challenging when considering the device energy consumption.

Corresponding Author: Chi Harold Liu. This work is financially sponsored by National Natural Science Foundation of China (Grant No. 61300179).

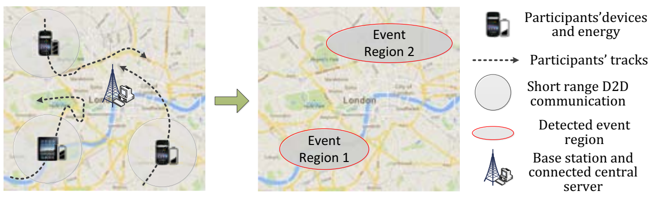
Event Region 2
Event Region 1
London
Participants'devices and energy
Participants' tracks
Short range D2D communication
Detected event region
Base station and connected central server

Fig. 1. The considered participatory sensing scenario for dynamic event detection.

There are mainly two categories of approaches to tackle the aforementioned problem. The first approach is that the participants transmit the collected raw data to the CS for centralized processing. For example, Sasaki *et al.* in [5] construct an earthquake reporting system where 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. However, in such a centralized algorithm, the communication overhead of uploading data can cost significantly to both the infrastructure and participants. The other approach is to process the data on each participant's smart device *distributively*. The authors of [6] present a distributed system that aims at classifying different application-specific events. Each node extracts some features out of its sensory data. By exchanging feature vectors with its neighboring nodes, each node classifies the event locally. However, this method can only be applied on small networks for validation. Using pattern matching techniques is another new trend for dynamic event detection. It could be applied distributively on each participant as in [7], or applied in the CS [8]. However, in some cases, the exemplary training data for events is very hard to generate.

Despite the complicated essence of an event, the data model used in experiments are sometimes over-simplified. For example, the authors of [9] generates their data by normal distributions with fixed parameters. In [10], the sensor fault is modeled by a uncorrelated Bernoulli random variable. Therefore, real world based simulations are needed to verify the performance of proposed approaches.

All the above research activities fail to consider the energy conservation problem, since participants tend to refuse to cooperate if an application consumes too much energy (or resources) from their devices. In this regard, 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 Higuchi *et al.* in [13] propose an energy-efficient information diffusion protocol for mobile crowd sensing. However, they have not built up mathematical foundations to establish the synergy between energy consumption and detection accuracy, in a distributed manner, as the central theme of our work in this paper.

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

1) We propose a centralized event detection algorithm that makes use of a *Minimum Cut* (Min-cut) algorithm as the performance benchmark.
2) We propose a distributed, energy-efficient event detection framework, in which the participants calculate events' regions by negotiating with their neighbors.
3) We propose a utility function from the solution of an optimization problem to embed in the distributed framework. The effectiveness and flexibility of the proposed algorithms are extensively evaluated by real-trace driven experiments.

The rest of the paper is organized as follows. Section II establishes formal model of our system. Section III describes the Min-cut based centralized event detection algorithm. Section IV describes our proposed distributed event detection framework and utility function. Section V extensively evaluates the performance of the proposed strategy by real trace-driven simulations, and finally Section VI concludes the paper.

II. System Model

We model the sensing region as a 2D map, denoted as $\mathcal{M}$, and within the region there are a set of 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. We assume that the system runs continuously for a long period of time, which contains many detection cycles. Each detection cycle consists of three phases: *Event Monitoring*, *Information Transmission* and *Central Processing*. That is, the participants collect data from the environment, perform some local computation if needed, and then transmit the necessary data to a CS server for final processing.

Formally, we define participants as a set $\mathcal{P} \triangleq \{i | i = 1, 2, \ldots, P\}$, where P is the size. For the rest of the paper, we refer to a participant i and his/her associated collective attributes together as a *participant*, or simply a *user*. These attributes are denoted by a tuple $p_i = (x_i, y_i, v_i, e_i)$, where x_i and y_i are a participant's coordinates, v_i is the sensory reading of his/her device, and e_i is the remaining energy level. Each user's device has an initial energy level e_0. We assume the participants move around in $\mathcal{M}$ randomly. A participant has necessary computing ability, and all participants have a common, but tunable communication range δ.

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	The coordinates, sensory reading and remaining energy 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 $G(\mathcal{P}, \mathcal{E})$
$\mathcal{E}_c$	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_0	Initial energy of each 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

For centralized algorithms, 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} denotes the edge between users i and j). Besides, the weight w_{ij} for each edge $l_{ij} \in \mathcal{E}$ is calculated as: $w_{ij} = \exp\left(-|v_i - v_j|/d(i,j)\right)$, where $d(i,j)$ denotes the distance between two participants i and j. It reflects the change of sensory readings between two neighboring users.

For distributed algorithms, 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 neighbor $j \in \mathcal{N}(i)$ is associated with a utility value U to explicitly represent the benefit of choosing user i. In addition, the *nearest* neighbor of a participant i is defined as:

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

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, the 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, etc.

Important notations used in this paper are listed in Table I.

III. Min-cut Based Centralized Approach

After the participants collect data from the environment, the raw data are transmitted to the CS for processing. Since the data transmission methods have been thoroughly studied in

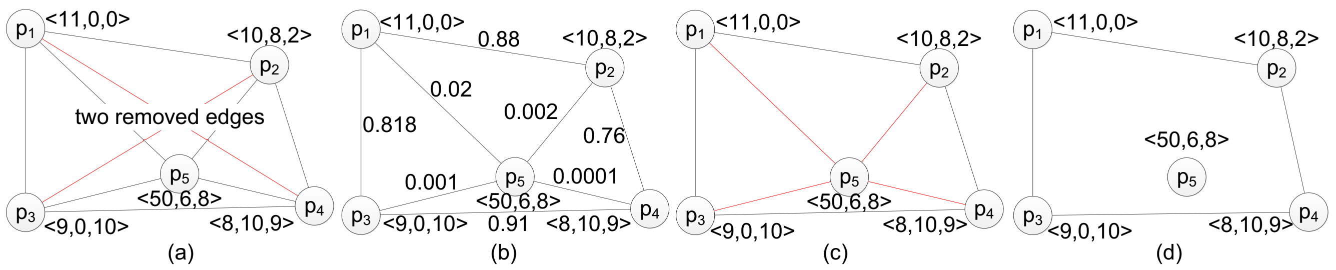

Fig. 2. An example of the Min-cut based algorithm. (a) Connect every two vertexes, (b) remove edges and calculate weights, (c) recognize edges to be removed, and (d) final detection result.

many previous research activities, we focus on the processing part in this paper. Besides, we only need to focus on one detection cycle. The rest of cycles have identical participant actions.

In $G(\mathcal{P}, \mathcal{E})$, our proposal's core step is to use an improved Min-cut algorithm [14] to find a subset $\mathcal{E}_\text{c} \subset \mathcal{E}$ that connect two neighboring participants *within and out of the area* of an event. Then all users that are in an event region can be separated out.

As an example, Fig. 2 shows the steps of how the participant(s) in an event region is(are) recognized in a mini-network with five participants $p_1 \sim p_5$. Obviously, with its abnormally high reading, sensor p_5 is identified in an event region. Thus edges p_1p_5, p_2p_5, p_3p_5, and 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. 2(c). Once these edges are removed, the participants in an event region (only p_5 in this case) are identified (see Fig. 2(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, every two participants are connected in the map, as shown in Fig. 2(a).

Step 2: Edge removal. To reserve only edges connecting neighboring users, those between non-adjacent users are removed. The remaining edges consist a new set $\mathcal{E}'$.

Step 3: Calculating edge weights. The weight w_{ij} for each edge $l_{ij} \in \mathcal{E}'$ is calculated.

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

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

When no more event area can be detected, the users in $\mathcal{P}_\text{out}$ are connected by edges in $\mathcal{E}_\text{c}$, and the users in $\mathcal{P}_\text{in}$ are isolated. These are exactly the participants that fall in the region of events. This algorithm does not assume a pre-defined threshold for events. Besides, it can detect multiple event regions. Moreover, by traversing all possible edges between any two users, this algorithm give precise detection result.

However, despite its high precision, this algorithm consumes much computational resources and runs slowly, which can be observed in the simulation section. Besides, it does not scale well with the size and structure of the network due to the communication bottlenecks and energy expenses.

IV. Distributed Event Detection Approach

Different from centralized algorithms, in a distributed approach, participants not only collect data, but also perform certain local processing, and then transmit 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 decision locally in an optimal way. 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 as follows. While each user can detect events according to the predefined threshold, many participants tend to provide similar detection results due to their similar geographical locations. Accordingly, we would like to group participants with "similar" neighbors, so that they are very likely to yield the same result during detection. Thus when one user is monitoring the environment, its *companions*, i.e. the other members in its group, do not have to monitor in the same cycle. Unless otherwise specified, for the rest of the paper we assume that each user can have at most one companion; and we shall study the impact of multiple companions during simulations. In this process, a user chooses its companion according to their pre-defined utilities. A proper utility definition is obviously a vital part in our detection framework, and it will be theoretically discussed in detail later.

At the beginning of each detection cycle, each user in region $\mathcal{M}$ selects its nearest neighboring users, referred as the "linking petition" to this neighbor. We refer to the initially chosen neighbor of participant i as its *desired user*. Conversely, i is a *petitioner* of its desired user. Each user has no prior

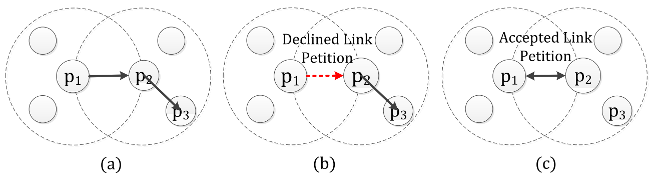

Fig. 3. An example of the conflict situation in the distributed framework.

knowledge of its neighbors' initial decisions before making its own decision, so it is highly possible that a user makes an improper decision.

Fig. 3 shows an example. We focus on users p_1 and p_2 in this case. The dash line circles denote their maximum transmission ranges. They both have some neighbors within the range. In Fig. 3(a), both users make initial decisions after computing their neighbors' utilities. However, user p_1 chooses p_2 whereas p_2 chooses its another neighbor p_3, that creates a conflict situation. Our solutions is that: if participant p_1 finds its desired user p_2's desired user is not p_1 itself, p_1 gives up its petition. Therefore, in Fig. 3(b), the linking petition of user p_1 is declined by p_2 after their initial decisions are exchanged. Fig. 3(c) shows another scenario where user p_1 and p_2 choose each other as initial decisions, hence form a group successfully. Below is a formal description of our algorithm.

Step 1: Exchange user information. Each user sends its sensory data to its neighbors, and then computes their utilities.

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

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

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

Step 5: Exchange petition decision. Each user notifies its petitioner whether it is accepted. Then, each user finally decide whether it should form a group with its desired user.

Step 6: Make sleep-wake decision. Each user decides whether itself should maintain sleep state in this cycle. If it is not in a group, it stays awake, otherwise whoever has most residual energy in this group stays awake.

After these steps, participants start to monitor the environment. If any event is detected, the corresponding users send alerts to the CS that contains information as short as 1 bit.

We next analyze the running time of our proposed algorithm, consisting of the time to complete: (a) the user or decision information exchange in Step 1, 3, 5, and (b) the decision-making process in Step 2, 4, 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. Furthermore, in the decision-making steps, the algorithm just traverses the information table of neighbors in the memory and find the proper one, which also takes constant time. In other words, the distributed approach is scalable whose running time is independent of network sizes.

B. Energy-Efficient Utility Definition

To complete our proposed distributed algorithm, a definition of the utility is important. For example, a simple Random-neighbor strategy could be applied, which means that a participant randomly chooses a neighbor for initial decision. However, to obtain a good metric, we need to build up synergies between detection accuracy and energy consumption.

We assume that each participant has an average sensory reading $\bar{v}_i(t)$ up to cycle t. An intuitive idea is that a user picks the neighbor that has the most similar average reading with itself. We calculate the *target accuracy* of neighbor k of any user i as : $T_i(k,t) = \exp\big(-|\bar{v}_i(t) - \bar{v}_k(t)|\big)$. However, when a participant's energy consumes, the detection accuracy is also decreased. Thus a neighbor with more remaining energy should be given higher priority to join the detection task. Therefore, for user i, we define its *accuracy* of selecting neighbor k at cycle t as: $\omega_i(k,t) = \exp\big(-|v_i(t) - v_k(t)|\big) \times \frac{e_k(t)}{e_0}$. Suppose 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 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 prove 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 factor of time.

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

$$\begin{aligned} &\max f(A_i) = \sum_{k=1}^{m} T_i(k).\log(A_i(k)), \\ &s.t.\ \sum_{k=1}^{m} A_i(k) \le C, \end{aligned} \tag{3}$$

where C is a positive constant that upper bounds the total average accuracy of all neighbors of user i. 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:

$$L = \sum_{k=1}^{m} T_i(k).\log(A_i(k)) - \lambda_i.\left(\sum_{k=1}^{m} A_i(k) - C\right). \tag{4}$$

The first order (necessary) optimality condition for (4) is:

$$\triangledown L = 0 \text{ and } \lambda_i\left(\sum_{k=1}^{m} A_i(k) - C\right) = 0. \tag{5}$$

Since the constraint is binding and $\lambda \neq 0$, the first part in (5) could be solved as: $T_i(k)/A_i^*(k) = \lambda_i$. 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 (3) converges to the optimal solution.

Lemma 2: If user i uses the following utility:

$$U_i(k,t) = T_i(k)\frac{\omega_i(k)}{A_i(k)}, \tag{6}$$

it maximizes the objective function in (3) iteratively.

Proof: Since the objective function in (3) is convex, the sufficient and necessary condition of optimality for this problem is: $\bigtriangledown f|_{\boldsymbol{A_i}}.(\boldsymbol{A_i} - \boldsymbol{A_i^*}) \leq 0$, where $\boldsymbol{A_i}$ could be any arbitrary energy consumption vector. This equation could be further broken into two parts:

$$\sum_{k=1}^{m} T_i(k)\frac{A_i(k) - A_i^*(k)}{A_i(k)}, \tag{7}$$

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

$$\sum_{k=1}^{m} T_i(k)\frac{E[\omega_i(k)]}{A_i(k)} - \sum_{k=1}^{m} T_i(k)\frac{E[\omega_i^*(k)]}{A_i(k)}, \tag{8}$$

where symbol E means the expectation of target accuracy for a series of cycles. Then maximizing the following will maximize the second part of (8):

$$\max_{\omega_i} T_i(k)\frac{\omega_i(k)}{A_i(k)}. \tag{9}$$

Since e_i in the first part in (8) is not optimal as in (9), the second part must be greater than the first term in (8). So this equation holds. That is, use the metric in (6) maximizes the objective function in (3). ■

In practical applications, taking the factor of time into consideration, (6) could be used as:

$$U_i(k,t) = T_i(k)\frac{\omega_i(k,t-1)}{A_i(k,t-1)}. \tag{10}$$

V. Performance Evaluation

A. Simulation Settings

We validate our proposed algorithms with the Microsoft Research Asia GeoLife data set [15]. It contains 182 volunteers' trajectories in Beijing for three consecutive years. We find a $200 \times 500m^2$ region that is of high movement density.Based on this data set, we consider a noise level monitoring application by participatory sensing. We use the number of visits on each area to indicate the possible noise level. The communication range of each participant is 10 meters. We also employ the energy dissipation model in [16], where the cost to transmit a L-bit message is:

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

where d denotes the distance between two users, ε_{tx} is the energy dissipation per bit to run the transmitter circuit, ε_{fs} and ε_{mp} are transmitter amplifiers, $d_0 = \sqrt{\varepsilon_{\text{fs}}/\varepsilon_{\text{mp}}}$. We adopt the same parameter settings in [16]: $\varepsilon_{\text{tx}} = 50$nJ/bit, $\varepsilon_{\text{fs}} = 10$pJ/bit/m^2 and $\varepsilon_{\text{fs}} = 0.0013$pJ/bit/m^4. The energy dissipated in event monitoring per round is 0.001mJ, and the initial energy of each user is set to 5mJ.

We assess the performance of our proposed Min-cut based centralized algorithm and distributed approach, by comparing with the *SVM-based* centralized algorithm and our proposed distributed algorithm using the Random-Neighbor strategy. We treat the detection problem as a pattern matching one. First, a SVM model is trained in the CS with prepared exemplary data, that contains participants' information and labels showing whether they are in an event region or not. Second, the participants transmit data back to the CS for classification.

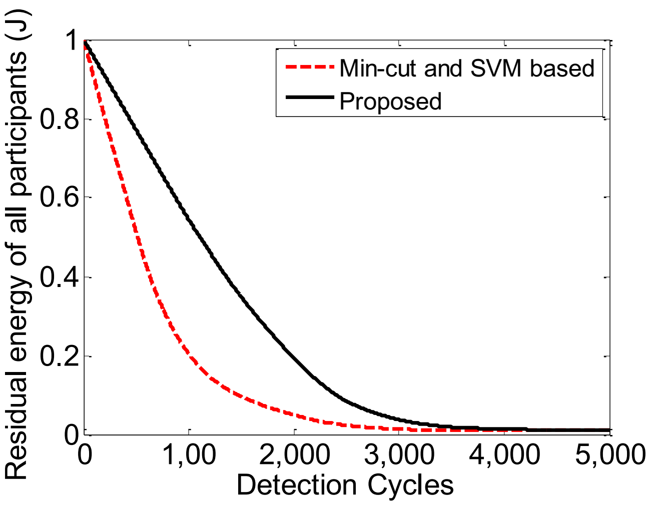

Fig. 4. Network lifetime performance of the proposed utility.

TABLE II
Running time of three algorithms with different user number.

No. of participants	100	200	300	400	500
Min-cut based (mins)	6″	1′20″	5′58″	14′46″	57′13″
SVM based (s)	0.0435	0.0441	0.0447	0.0458	0.0472
Proposed (s)	0.0045	0.0095	0.0143	0.0191	0.0239

B. Results and Discussions

First we 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. On the contrary, the distributed approach can run 9.67 times faster than the SVM-based algorithm. We can also observe that with the state-of-the-art solvers and optimized codes, the running time of SVM does not change dramatically with the increase of number of participants in this order of magnitude.

Fig. 4 shows the total energy consumption with 200 users. When a user has less than 1% energy left, it is deemed inactive. When the detection process continues, the distributed algorithm consumes much less energy. Fig. 5(b) shows the network lifetime. With the increase of number of participants, the network lifetime achieved by Min-cut based algorithm barely grows, while the distributed achieves much higher and

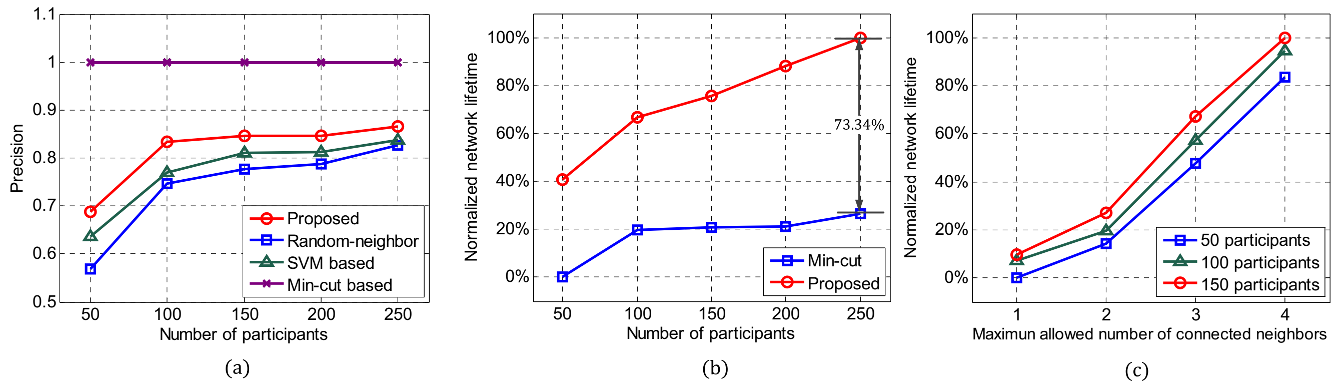

Fig. 5. (a)-(b) Precision and network lifetime with different number of participants. (c) Network lifetime performance with different maximum allowed number of connected neighbors.

increases significantly. Here we define a network (or sensing region) as not functional, when 90% users are inactive. From the figure, we observe that the network lifetime is prolonged 73% with distributed algorithm than with Min-cut approach.

Since the centralized algorithm detects events more accurately, we use its detection result as the benchmark. As for the distributed algorithm, we investigate its performance with two utilities: our proposed one in (10), and the random-neighbor strategy. We use detection *precision* as merit to evaluate their performance, calculated as follows. Suppose TP is the number of detected event node that are true event node, FP is the number of detected event node that are actually non-event node, the precision (P) is then defined as: $P = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$. Our proposed solution can maximally achieve 86% precision, 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.

As stated in Section IV, a participant always selects its nearest neighbor to form a group. Here we aim to explore the impact of using different 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. 5(c) shows the network lifetime with different number of participants. It is clear that the more neighbors one user can choose, the more enduring a network is. Despite the number of participants, if K is set as 4 instead of 1, the network lifetime is prolonged 85% on average.

VI. Conclusion

Dynamic event detection by participatory sensing is a promising research direction. In this paper, we proposed two event-detection algorithms: a Min-cut based centralized approach, and more importantly a distributed detection framework. In the distributed algorithm, an optimization problem is formalized and solved to derive an optimal utility that ensures the detection precision and energy-efficiency of the algorithm. Extensive experimental results, based on a real-trace driven data set, show that our proposed distributed algorithm detect events fast, accurately and energy-efficiently.

References

[1] 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 *ACM MobiSys'09*, 2009, pp. 165–178.

[2] 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 *ACM SIGKDD'13*, 2013, p. 15.

[3] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, “Parknet: drive-by sensing of road-side parking statistics,” in *ACM MobiSys'10*, 2010, pp. 123–136.

[4] 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 *ACM/IEEE IPSN'10*, 2010, pp. 105–116.

[5] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in *ACM WWW'10*, 2010, pp. 851–860.

[6] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller, “A system for distributed event detection in wireless sensor networks,” in *ACM/IEEE IPSN'10*, 2010, pp. 94–104.

[7] 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, 2012.

[8] W. Xue, Q. Luo, L. Chen, and Y. Liu, “Contour map matching for event detection in sensor networks,” in *ACM SIGMOD'06*, 2006, pp. 145–156.

[9] Z. Zhou and G. Qu, “An energy efficient adaptive event detection scheme for wireless sensor network,” in *IEEE ASAP'11*, Sept 2011, pp. 235–238.

[10] B. Krishnamachari and S. Iyengar, “Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks,” *IEEE Trans. Comput.*, vol. 53, no. 3, pp. 241–250, March 2004.

[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, 2010.

[13] T. Higuchi, H. Yamaguchi, T. Higashino, and M. Takai, “A neighbor collaboration mechanism for mobile crowd sensing in opportunistic networks,” in *IEEE ICC'14*, June 2014, pp. 42–47.

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

[15] Y. Zheng, X. Xie, and W. Ma, “Geolife: A collaborative social networking service among user, location and trajectory,” *IEEE Data Engineering Bulletin*, vol. 33, no. 2, pp. 32–40, 2010.

[16] 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, 2002.