Energy-Efficient Collaborative Sensing with Mobile Phones

Xiang Sheng*, Jian Tang* and Weiyi Zhang†
*Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY 13244.
†AT&T Labs Research, Middletown, NJ 07748.

***Abstract*—Mobile phones with a rich set of embedded sensors enable sensing applications in various domains. In this paper, we propose to leverage cloud-assisted collaborative sensing to reduce sensing energy consumption for mobile phone sensing applications. We formally define a minimum energy sensing scheduling problem and present a polynomial-time algorithm to obtain optimal solutions, which can be used to show energy savings that can potentially be achieved by using collaborative sensing in mobile phone sensing applications, and can also serve as a benchmark for performance evaluation. We also address individual energy consumption and fairness by presenting an algorithm to find fair energy-efficient sensing schedules. Under realistic assumptions, we present two practical and effective heuristic algorithms to find energy-efficient sensing schedules. It has been shown by simulation results based on real energy consumption (measured by the Monsoon power monitor) and location (collected from the Google Map) data that collaborative sensing significantly reduces energy consumption compared to a traditional approach without collaborations, and the proposed heuristic algorithm performs well in terms of both total energy consumption and fairness.**

***Index Terms*—Mobile phone sensing, energy-efficiency, opportunistic sensing, collaborative sensing, scheduling.**

I. Introduction

Mobile phones have evolved as key electronic devices for communications, computing and entertainments, and have become an important part of people's daily life. Most of current smart phone models (such as iPhone 4S, HTC's Android phones, etc) are also equipped with a rich set of embedded sensors such as camera, GPS, WiFi/3G/4G interfaces, accelerometer, digital compass, gyroscope, microphone and so on [6]. Moreover, external sensors can also be connected to a mobile phone via its Bluetooth interface. These sensors can enable sensing applications in various domains [6] such as environmental monitoring, social network, healthcare, transportation, safety, etc. For example, researchers from University of California at Los Angeles (UCLA), developed a mobile phone sensing application, called PEIR (Personal Environmental Impact Report) [13], which uses location data sampled from mobile phones everyday to calculate personalized estimates of environmental impact and exposure. A noise map facilitates monitoring of environmental noise pollution in urban areas. Researchers from University of New South Wales and Portland State University developed a participatory urban noise mapping system called Ear-Phone [16], to create noise maps for different areas. In addition, *www.sensorly.com* is a website which offers free access to 100% community powered coverage maps for various wireless networks (3G/4G/WiFi). Their mapping crowd collects data everyday using a free application on mobile phones, which produces a true picture of carriers' coverage. More mobile phone sensing applications have been introduced in [6].

Even though these applications are very attractive, performing sensing tasks using a mobile phone may consume significant amount of energy. Therefore, without carefully managing very limited energy resources on mobile phones, mobile users may end up with an awkward situation after performing a few sensing tasks, in which phones are out of battery when they are needed to make phone calls. There is a large space for energy savings on a mobile phone. In this work, we study how to minimize sensing energy consumption such that mobile phones can undertake sensing tasks, and in the meanwhile, they can still fulfill their regular duties, such as phone calls, emails, etc.

There are two mobile phone sensing paradigms [6]: 1) *Participatory Sensing:* users actively engage in sensing activities by manually determining how, when, what, and where to sense. 2) *Opportunistic Sensing:* Sensing activities are fully automated without user involvement. In this work, we focus on opportunistic sensing applications (e.g., environmental noise sensing [16], wireless signal sensing [20], etc). We aim to develop general (application-independent) methods to control the sensing procedure with the objective of minimizing sensing energy consumption. A commonly used method is to make every mobile phone sense periodically (every x seconds). This method is obviously not efficient because if this method is used, many redundant data reports may be produced for a target region by a large number of users which happen to show up in that area. Redundancy (i.e., the number of times a user senses) can be reduced and energy-efficiency can be improved by using a coordinator to control sensing activities of users such that those mobile phones sense collaboratively to produce just enough data reports for the application. To this end, we propose to use a cloud-assisted collaborative sensing approach. Cloud computing has evolved as an important computing model, which can be leveraged to assist mobile phone sensing by using servers in a cloud to not only handle data reports from mobile phones but also collect mobility and location information from mobile phones, calculate the best sensing schedule and tell them when/where to sense. Those sensors that are not needed for sensing can be turned off or work in a low power mode.

Participatory sensing and opportunistic sensing have been studied by a few recent works [6], most of which were focused on the design and implementation of a particular mobile phone sensing application. However, we address fundamental and general sensing scheduling and coverage issues for opportunistic sensing without targeting a particular application. In addition, algorithms proposed for mobile sensor networks [18], [25], [26] cannot be applied here since they usually assume that mobility of sensors can be controlled to provide certain sensing coverage. However, in mobile phone sensing scenarios, sensors' mobility is usually uncontrollable. In this paper, we study optimization problems related to energy-efficient collaborative sensing with mobile phones. *Our contributions are summarized in the following:*

1) Assuming knowing each user's moving trajectory in advance, we present a polynomial-time algorithm to obtain minimum energy sensing schedules. Even though this assumption may not be realistic, the obtained solutions can be used to show potential energy savings that can be brought by using collaborative sensing in mobile phone sensing applications, and can also serve as a benchmark for performance evaluation. We also address individual energy consumption and fairness by presenting an algorithm to find fair energy-efficient sensing schedules.
2) We present practical and effective heuristic algorithms to find energy-efficient sensing schedules under realistic assumptions.
3) We present simulation results based on real location (collected from the Google Map) and energy consumption (measured by the Monsosn power monitor [12]) data to show that collaborative sensing significantly reduces energy consumption compared to a traditional approach without collaborations, and the proposed heuristic algorithm perform well in terms of both total energy consumption and fairness.

To the best of our knowledge, we are the first to present theoretically well-founded and practically useful algorithms to show the benefits of using collaborative sensing in mobile phone sensing applications. The rest of this paper is organized as follows. We discuss related works in Section II. We describe the system model and formally define the problems in Section III. The proposed algorithms are presented in Section IV. We present the simulation results in Section V and conclude the paper in Section VI.

II. Related Work

Mobile phone sensing has recently attracted extensive research attention from both academia and industry due to its attractive applications. Besides applications mentioned above, there are also mobile phone sensing projects on social networking [11], health and well being [3] and many other areas [6]. For example, the CenceMe [11] system represents the first system that combines the inference of the presence of individuals using mobile phones with sharing of this information through social networking applications such as Facebook and MySpace. The UbiFit Garden [3], a joint project between Intel and the University of Washington, captures levels of physical activity and relates this information to personal health goals by presenting feedbacks to the user. A comprehensive review of these applications can be found in a recent survey paper [6].

Sensing and coverage related issues have been studied in the context of mobile phone sensing recently. In [17], Reddy *et al.* proposed a network service architecture for participatory sensing and described challenges in network coordination services, attestation mechanisms and participatory privacy regulation mechanisms. In [9], the authors presented the design, implementation and evaluation of the Jigsaw continuous sensing engine for mobile phones, which balances the performance needs of an application and resource demands. Jigsaw comprises a set of sensing pipelines for accelerometer, microphone and GPS sensors, which are built in a plug and play manner to support resilient accelerometer data processing, smart admission control and adaptive pipeline processing. The authors of [14] presented the design, implementation and evaluation of several techniques to optimize the information uploading process for continuous sensing on mobile phones. The paper [7] studies economic models of user participation incentive in participatory sensing applications. In [5], the authors presented Zoom, a multi-resolution tasking framework for crowd-sourced geo-spatial sensor networks. Zoom allows users to define arbitrary sensor groupings over heterogeneous, unstructured and mobile networks and assign different sensing tasks to each group.

Only few recent works addressed collaborative sensing with mobile phones. In [10], the authors presented analytical results on the rate of information reporting by uncontrolled mobile sensors needed to cover a given geographical area, and demonstrate the feasibility of using existing software and standard protocols for information reporting and retrieval to support a large system of uncontrolled mobile sensors using a testbed. In [23], the authors introduced mechanisms for automated mapping of urban areas that provide a virtual sensor abstraction to applications. They also proposed spatial and temporal coverage metrics for measuring the quality of acquired data. In [22], the authors proposed a protocol, Aquiba, that exploits opportunistic collaboration of pedestrians. Its performance was studied via simulations.

Collaborative sensing has been well studied for mobile sensor networks. In [26], Zhou *et al.* considered how to deploy mobile sensors into an existing sensor network to enhance its connectivity and coverage, and presented a dynamic programming based algorithm under the assumption that each sensor is equipped with GPS. Several distributed algorithms were presented for a sensing coverage problem in [25], which do not need any location and distance information. In [18], Saipulla *et al.* explored the fundamental limits of sensor mobility on barrier coverage and presented a sensor mobility scheme that constructs the maximum number of barriers with minimum sensor moving distance.

The differences between our work and these related works are summarized as follows: 1) Unlike papers that were focused on particular mobile phone sensing applications [3], [11], [13], [16], [20], we aim at designing general (instead of application-specific) approaches to enable energy-efficient sensing with mobile phones. 2) The optimization problems considered in related works [5], [7], [9], [10], [14], [17], [22], [23] are mathematically different from the problems considered here. 3) Closely related works [22], [23] presented heuristic algorithms that cannot provide performance guarantees. We, however, present algorithms to produce optimal solutions, which can be used to justify the use of collaborative sensing in opportunistic sensing applications and show the potential energy savings quantitatively. This is the major contribution of this paper. 4) The algorithms presented in [18], [25], [26] for mobile sensor networks (in which sensor mobility can be controlled to achieve certain sensing coverage) cannot be applied here because the mobility of mobile phones is usually uncontrollable.

III. Problem Definition

First, we summarize the major notations in Table I.

TABLE I
Major Notations

$G(V,E)$	The VSG and its vertex and edge sets.
M	The number of roads in the target region.
N	The number of mobile users/phones
S_i	The sensing schedule of user i
T	The deadline of the given sensing task.
w_i	The energy needed to sense once using user i's phone.
Γ_i	The moving trajectory of user i

We consider a mobile phone sensing system with multiple mobile users, each of which carries a mobile phone equipped with sensors. The mobility of each mobile user cannot be controlled. However, mobile users' movements are highly restricted by roads, i.e., a vehicle or a person can only move along roads and turn at intersections. The movements of a mobile user i can be characterized using a trajectory Γ_i which is a set of 3-tuples (i, t_i, loc_i) and each of them gives the location of user i at time t_i. The more 3-tuples there are in the trajectory, the more accurately it can characterize the movements of mobile user i.

Again, we focus on the opportunistic sensing scenario in which sensors on each mobile phone automatically perform sensing tasks without user involvement. Each sensor is assumed to have a sensing range of r, which basically means if a user sense at a location loc_x and obtain a reading, then there is no need to sense again in any location within the disk with the origin at loc_x and a radius of r since the readings will be similar. Sensing target areas are roads in a given region, which are assumed to be narrow chains. The width of a road is assumed to negligible because r is usually larger or much larger. We consider regular roads which are roughly straight roads. Those irregular roads can be treated as a sequence of regular roads. If a user is on a road, it is assumed to cover a segment (b, c) with a length of $2r$. We can be more or less conservative on coverage by setting the value of r to a smaller or larger value. In the following, we will use *user*, *phone* and *sensor* interchangeably.

Given a sensing application, multiple servers are set up in a cloud to coordinate sensing activities. Servers are assumed to periodically exchange information to have a consistent view of mobile phones in the system. Each mobile phone can exchange information with one of the servers using its wireless interface. Our approach is to use servers to gather information from mobile phones, determine when/where to sense for each phone, send the sensing schedule to mobile phones and then collect sensed data reports from them.

Every 3-tuple in a trajectory can be imagined as a *virtual sensor*. If trajectories of all users are given, then we will have a large network of *virtual sensors* by combining all 3-tuples in trajectories. A sensing schedule S is a collection of virtual sensor sets, i.e., $S = \bigcup_{i=1}^{N} S_i$, where $S_i \subseteq \Gamma_i$. $|S_i|$ gives the number of times user i (mobile phone of user i) senses. Note that performing a common sensing task may consume different amount of energy on different phones. For example, energy consumption of a WiFi scan on three popular Andorid-based smart phones can be found in Table II.

Since energy efficiency is the primary design goal of this work, we want to minimize total energy consumed in the whole sensing procedure. So we define the following optimization problem.

Definition 1 (MECSS): Given a region, M roads in the region, N mobile users, a deadline T and the moving trajectory Γ_i of each user $i \in \{1, \cdots, N\}$ before the deadline, the **Minimum Energy Collaborative Sensing Scheduling (MECSS)** problem seeks a sensing schedule $S_i \subseteq \Gamma_i$ for each user i, such that its total energy consumption $\sum_{i=1}^{N} w_i |S_i|$ (where w_i is the energy needed to sense once with the mobile phone of user i) is minimized subject to the constraint that the roads in the given region are fully covered before the deadline T.

However simply minimizing the total energy consumption may lead to unfair utilizations of mobile phones, some users' phones are heavily used for sensing and other users' phones are lightly utilized or not utilized at all. A similar issue has been shown by previous works [21] for wireless mesh and sensor networks: simply maximizing network throughput leads to severe unfairness on users' individual throughput. Therefore, we try to improve fairness by only considering those sensing schedules with the min-max number of user sensing times. We call such schedules *min-max fair* sensing schedules. We also study the *Fair Energy-efficient Collaborative Sensing Scheduling (FECSS)* problem which seeks a min-max fair sensing schedule $S_i \subseteq \Gamma_i$ for each user i, such that its total energy consumption is minimized.

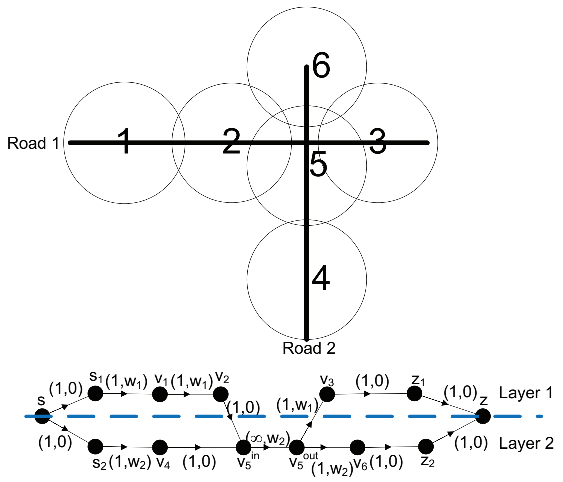
Assuming knowing the trajectory of each mobile user in advance, we present algorithms to solve these sensing scheduling problems optimally, which are presented in Section IV-A. It may be argued that these assumptions are not realistic since it is hard to precisely predict how users move in the future and mobile users need to turn on the GPS devices on their mobile phones to obtain precise locations, which, however, are energy-hungry [8] (a GPS device usually consumes much more energy than other sensors). However, the optimal solutions can be used to show energy savings that can potentially be achieved by collaborative sensing and they can serve as a benchmark for performance evaluation, i.e., can be used to find out how far a sensing schedule produced by a practical heuristic algorithm is away from the optimal. Therefore, it makes sense to present the optimal algorithms. We also present two practical and simple heuristic algorithms in Section IV-B to find energy-efficient sensing schedules for mobile users, which do not need moving trajectories beforehand or precise locations.

IV. Collaborative Sensing Algorithms

A. Optimal Algorithms

In order to solve the MECSS problem defined above, we first introduce a graph model, Virtual Sensor Graph (VSG) $G(V, E)$, to assist computation. As mentioned above, the moving trajectory of each mobile user is assumed to be known and each 3-tuple (i, t_i, loc_i) in a trajectory can be viewed as a *virtual sensor*. The sensing scheduling problem is actually to find a subset of virtual sensors to cover the roads in the target region. This graph is a multi-layer directed graph and each layer corresponds to a road L in the target region. Every vertex corresponds to a virtual sensor v_j (associated with user i). Let (b_j, c_j) and $(b_{j'}, c_{j'})$ be segments of L covered by virtual sensors v_j and $v_{j'}$ (that are on L), respectively. There is a directed edge from v_j to $v_{j'}$ if $b_j < b_{j'} \leq c_j < c_{j'}$ (i.e., their coverage areas overlap). Its capacity and cost are set to 1 and $w_{i'}$ (virtual sensor $v_{j'}$ is associated with user i'), respectively. For a virtual sensor (vertex) on multiple roads (e.g., virtual sensors at inter-sections are on two or more roads.), we arbitrarily pick a corresponding layer (road) to place it in G. Moreover, for each virtual sensor v (associated with user i) covering segments on multiple roads, we have to create a pair of vertices (v^{in}, v^{out}) to represent it in G, and there is a directed edge from v^{in} to v^{out} whose capacity and cost are set to ∞ and w_i respectively. We call such virtual sensors *cross-road* virtual sensors and those edges *intra-vertex* edges. There is a directed edge from v^{out} to another vertex u or from another vertex u to v^{in} if the aforementioned condition is met. The capacities and costs of all incoming edges associated with a cross-road virtual sensor are set to 1 and 0 respectively. However, the capacity and cost of an outgoing edge (to vertex u associated with user h) associated with a cross-road virtual sensor are set to 1 and w_h respectively. Note that another vertex u here may be in the same layer or in a different layer. Therefore, edges associated with cross-road virtual sensors may cross layers.

In addition, we also add a virtual source s_m for each layer $m \in \{1, \cdots, M\}$ and edges from s_m to all the vertices whose corresponding virtual sensors cover the western/southern boundary of the corresponding road with their capacities and costs set to 1 and the energy cost of the user associated with its destination vertex, as well as a virtual sink z_m and edges from all the vertices whose corresponding virtual sensors cover the eastern/northern boundary of the corresponding road to z_m with their capacities and costs set to 1 and 0 respectively. In addition, there are an ultimate virtual source s and sink z. There are also edges from s to virtual sources in all layers with their capacities and costs set to 1 and 0, and edges from virtual sinks in all layers to z with their capacities and costs set to 1 and 0 too. A simple example in Fig. 1 is used to illustrate the graph construction. In this example, we have 6 virtual sensors and two roads. Virtual sensors $1, 2, 3$ are on road 1 and are assumed to be associated with user 1; and virtual sensors $4, 5, 6$ are on road 2 and they are assumed to be associated with user 2, as illustrated by the first sub-figure. The corresponding 2-layer VSG is given in the second sub-figure, in which the first number associated with each edge is its capacity and the second number is its cost. In this example, vertex v_5 corresponds to a cross-road virtual sensor which can be used to cover both roads 1 and 2. Intuitively, such virtual sensors should be fully leveraged to reduce sensing energy consumption. We present our optimal algorithm for the MECSS problem as follows.



6
Road 1
1
2
3
5
4
Road 2
s
(1,0)
s_1 $(1,w_1)$ v_1 $(1,w_1)$ v_2
(1,0)
$(1,w_1)$
v_3 (1,0) z_1
(1,0)
z
Layer 1
(1,0)
s_2 $(1,w_2)$ v_4 (1,0) v_5^{in}
(∞,w_2)
v_5^{out} $(1,w_2)$ v_6 (1,0) z_2
(1,0)
Layer 2

Fig. 1. Roads, virtual sensors and the corresponding VSG

Algorithm 1 The optimal MECSS algorithm

```
Step 1  Construct the VSG $G(V, E)$;
Step 2  Solve the LP relaxation of the ILP-MinE;
Step 3  **if** (No feasible solution)
            **output** ”There is no feasible solution!”;
        **else**
            **output** the corresponding sensing schedule;
        **endif**
```

Unknown decision variables:
1) f_e ($e \in E$): the amount of flow on link e.
2) $x_e = \{0, 1\}$ ($e \in E$): If $x_e = 1$, link e in G is selected; $x_e = 0$, otherwise.

ILP-MinE:

$$\min \sum_{e \in E} w_e x_e \tag{1}$$

Subject to:

$$\sum_{e \in E_s^{out}} f_e = M, \tag{2}$$

$$\sum_{e \in E_v^{out}} f_e = \sum_{e \in E_v^{in}} f_e, \qquad \forall v \in V \setminus \{s, z\}; \tag{3}$$

$$f_e \leq C_e, \qquad \forall e \in E; \tag{4}$$

$$x_e \begin{cases} \geq f_{e'}, \forall e \in E^{intra}, \forall e' \in E_e^{in}; \\ = f_e, \forall e \in E - E^{intra}. \end{cases} \tag{5}$$

In this algorithm, the LP relaxation of an Integer Linear Programming (ILP), ILP-MinE, needs to be solved to obtain optimal solutions. In the ILP, f_e and x_e $(e \in E)$ are both integer variables and E_v^{in}/E_v^{out} is the set of incoming/outgoing edges of vertex v on G. E^{intra} is the set of intra-vertex edges and E_e^{in} is the set of incoming edges associated with the source vertex of edge e on G. Once we obtain values for x_e by solving the corresponding LP relaxation (which are guaranteed to be 0 or 1), then we can figure out which virtual sensors should be selected for sensing (i.e., which user should sense at when and where). Specifically, if $e = (u, v)$ is a regular edge in G and $x_e = 1$, then the virtual sensor corresponding to its destination vertex v will be selected for sensing. If e is an intra-vertex edge and $x_e = 1$, then obviously the virtual sensor corresponding to e will be selected for sensing. We have the following proposition.

Proposition 1: Algorithm 1 optimally solve the MECSS problem in polynomial time.

Proof: The importance of the SVG lies in the fact that any feasible integer $s - z$ flow with a total flow amount of M (the number of roads in the target region) gives a feasible (in terms of coverage) sensing schedule. This is because our graph construction guarantees that every integer $s - z$ flow in a layer (which may include edges from different layers) corresponds to a sensing schedule that can fully cover the road corresponding to that layer. For example, in Fig. 1, an integer flow $(s, s_1, v_1, v_2, v_5^{in}, v_5^{out}, v_3, z_1, z)$ corresponds to a sensing schedule with virtual sensors $1, 2, 5$ and 3. Note that we introduce two virtual vertices in each layer that are used to deal with the case where multiple virtual sensors may be able to cover the head of a road. Setting corresponding edges' capacities to 1 ensures that fully covering each road once instead of covering a road more than once but leaving some other roads not fully covered.

In addition, the costs of most regular edges (except those associated with virtual vertices) are set to the energy cost of users (associated with their destination vertices). Then the selection of an edge $e = (u, v)$ basically means the corresponding virtual sensor (i.e., the virtual sensor corresponding to vertex v) is added to the sensing schedule. For those vertices corresponding to cross-road virtual sensors which may make contributions for covering of multiple roads, the costs of all the corresponding incoming cross-layer edges are set to 0 and the cost of the corresponding intra-vertex edge is set to the energy cost of the corresponding user. This way of assigning link costs along with Constraints (5) ensure that no matter how many roads can benefit from the coverage contributions made by using this vertex (virtual sensor), it is only counted once. The costs of those edges associated with virtual vertices are also assigned properly (e.g., $cost(e) := 1$, where $e = (s_i, v)$; while $cost(e') := 0$, where $e' = (v', z_i)$) such that by counting the total costs of selected edges, we can find out the total energy consumption. Hence, due to the way how the costs of edges in a VSG is assigned, we can claim that a minimum cost $s - z$ flow with a flow amount of M actually corresponds to a feasible (coverage-wise) sensing schedule with the minimum energy consumption.

By replacing variables x_e in the objective function with Constraints (5), we can see that solving the ILP-MinE is equivalent to solving a series of ILP, each of which has Constraints (2)–(4) and an objective function of $\min \sum_{e \in E - E^{intra}} w_e f_e + \sum_{e \in E^{intra}} w_e f_{e'}$ (where e' is one of incoming edge of the intra-vertex edge e); and then take the maximum of all objective values. Each such an ILP is a minimum-cost-flow-like problem, whose coefficient matrix is totally unimodular [24]. It is known that solving the LP relaxation of such an ILP problem automatically yields integral optimal solutions [24]. Furthermore, the ILP-MinE obviously includes polynomial numbers of variables and constraints. Therefore, the LP relaxation of the ILP-MinE can be solved by existing algorithms [1] in polynomial time, which can yield integral optimal solutions. This completes the proof. ■

The FECSS problem can be solved by an algorithm similar to Algorithm 1. Instead of solving the ILP-MinE, if we solve two following ILPs sequentially, then we can obtain a fair energy-efficient sensing schedule. Specifically, we first solve the ILP-Maxmin and obtain the min-max number of sensing times β. Because of Constraints (7) and the objective function, we can guarantee that for any feasible solution given by solving the ILP-Maxmin is min-max fair (according to our definition above). Next, we feed β to the ILP-FECSS as a parameter, which has the objective function of minimizing the total energy consumption and Constraints (7). Here, E_i is the set of edges associated with user i. Note that for a user with cross-road virtual sensors, only the corresponding intra-vertex edges are counted. Therefore, solving the ILP-Maxmin and ILP-FECSS in sequence can provide an optimal solution for the FECSS problem. We also used solutions generated by this algorithm as a benchmark for comparison in our simulation.

ILP-Maxmin:

$$\min \beta \tag{6}$$

Subject to: Constraints (2)–(5)

$$\sum_{e \in E_i} x_e \le \beta, \quad \forall i \in \{1, \cdots, N\}; \tag{7}$$

ILP-FECSS(β):

$$\min \sum_{e \in E} w_e x_e$$

Subject to: Constraints (2)–(5) and (7)

B. Practical Heuristic Algorithms

In this section, we present two practical heuristic algorithms. First, we do not assume the moving trajectory of each user is known; second, we do not assume that users use their energy-hungry GPS devices all the time. However, without knowing anything about mobile users, the only thing we can do is probably to let them sense periodically. Therefore, we do assume that users' moving directions and speeds can be detected and measured using some sensors (such as accelerometer and digital compass) on mobile phones and a method such as that introduced in [4]. Furthermore, the GPS device is assumed to be turned on right after a user initiates a sensing task to provide one or multiple reference locations for mobility prediction. It can certainly be (automatically or manually) turned off after necessary information is collected. In this way, a mobile user can keep track of where he/she is during the sensing procedure. In addition, every time a mobile user enters a new road segment (which can be detected by the mobile phone by measuring the distance travelled and detecting the direction change using an accelerometer and a digital compass), a short report message will be automatically sent to a server by his/her mobile phone. Note that in this section, a road segment is defined by two intersections, which may be different from the "road" (which may include a set of consecutive road segments) considered in the last section.

Both heuristic algorithms are used by a server to calculate a sensing schedule which will then be broadcast to mobile users. The first algorithm can be viewed as a realistic way to apply the optimal algorithms presented above. The basic idea is to sequentially use an algorithm presented above with partial trajectories that can be predicted to find out how to sense for the next certain period of time. We call this algorithm the *prediction-based* algorithm, which is presented as follows.

Algorithm 2 The prediction-based algorithm

```
Step 1  Predict users' moving trajectories (until the earliest
        time a user will enter a new road segment) according
        to their current locations and mobility information;
        Generate virtual sensors according to the predicted
        partial trajectories;
Step 2  Based on these virtual sensors, construct a connected
        VSG and apply the optimal MECSS or FECSS al-
        gorithm to calculate a new sensing schedule and
        broadcast it to mobile users;
Step 3  Update the target region by removing the road seg-
        ments that have been covered;
        Update the number of times each user has already
        sensed;
        if (Receive a report)
            if (Roads in the target region is fully covered
                or time is up)
                return ;
            else goto Step 1;
            endif
        endif
```

In this algorithm, we do NOT predict how a user will do in an intersection (make a turn, go straight, or even u-turn), which is hard. Instead, we apply an algorithm to predict how the user will move towards the intersection he is facing, from which we can obtain a partial trajectory (for each mobile user) that characterizes his movement from current location to wherever he will reach at the earliest time a user (himself or another one) will reach an intersection. Note that any prediction algorithm (e.g., an application-specific prediction algorithm) can be applied here. In the simulation, we used a simple but practical method, which assumes that the user will move towards the intersection he is facing without changing his direction or speed. The VSG constructed based on partial trajectories may not be connected. We simply connect disconnected components (if there are any) by connecting vertices on the edge to produce a connected graph. In Step 2, every time the same algorithm, the optimal MECSS or FECSS algorithm, is applied, however, the input changes over time because the algorithm needs to take account of the portions of roads that have been covered as well as the number of times each user has already sensed. In the simulation, we used the optimal FECSS algorithm. Obviously, the prediction-based algorithm does not always yield optimal solutions, however, we show that it works fine on average cases via simulations.

This second algorithm uses a function of a couple of sensing-related factors to make sensing decisions for mobile users. Hence, we call it the *function-based* algorithm, which is formally presented as Algorithm 3.

Algorithm 3 The function-based algorithm

```
Step 1  Generate virtual sensors according to the roads in
        the target region to make sure all the roads are fully
        covered;
Step 2  Select a minimum subset of virtual sensors that can
        cover all the roads in the target region using the
        MECSS algorithm and store them in $V_S$;
Step 3  while (1)
            if (Receive a report about a new road segment $L$)
                if ($V_S^L \neq \emptyset$)
                    Use a function to determine the number $J$ of
                      virtual sensors in $V_S^L$ that need to be used;
                      Notify the user to use the first $J$ virtual
                      sensors and remove them from $V_S^L$;
                endif
                if ($V_S = \emptyset$ or time is up) return ;
                endif
            endif
        endwhile
```

First, this algorithm generates virtual sensors on the roads in the target region to make sure all the roads are fully covered and tries to find a sensing schedule with a minimum subset of virtual sensors (i.e., a minimum number of sensing times) to cover all the roads in the target region. This can be easily done by applying the optimal MECSS algorithm described above with $w_i (i \in \{1, \cdots, N\})$ set to 1.

Again, we assume that when a user enters a new road segment L, it will notify the server with a report message. Then the server needs to determine how to make this user sense in this new segment. In the algorithm, $V_S^L \subseteq V_S$ is the subset of virtual sensors on L that are selected in the second step. The V_S^L is updated every time after some virtual sensors are used (i.e., some users used their sensors to sense at times and locations specified by these virtual sensors).

Our function-based algorithm can rather be considered a general optimization framework that uses a sensing-related function to determine how many times a user should sense in the road segment he/she enters. Any function can be used here, however, it may not lead to good performance. Here are some guidelines for designing a "good" function: 1) The function value should decrease with the number of times this user has already sensed for fairness purpose. 2) It should increase with the decrease of time left to perform the sensing task. 3) If no (or almost no) time left, $J = |V_L^S|$, where J is the value returned by the function. We suggest to use the following exponential function in the algorithm:

$$f(t, T, q, \overline{q}, c) = \lceil e^{-c\frac{q}{\overline{q}}\frac{t}{T}} |V_S^L| \rceil, \qquad (8)$$

where t and T are the time left to complete the sensing task and the deadline respectively; q and $\overline{q}$ are the number of times this user has already sensed and $\overline{q}$ is the average number of sensing times among all users. c is a tunable parameter. This function certainly satisfies the three requirements mentioned above and its values fall in the range $(0, |V_S^L|]$. Note that $|V_S^L|$ gives the number of virtual sensors left for use. Furthermore, the value of c can be set in a certain way to achieve a good tradeoff between coverage and fairness. Specifically, a smaller c can ensure coverage but may lead to unfair sensing schedule (some users' phones may be abused!); however, a larger c leads to fair sensing but may result in loss of coverage. We performed simulations to evaluate the performance of this function and to investigate what is the best value for the constant c, which will be discussed in details later. Certainly, some other application-specific factors may be brought into the function to (hopefully) improve the performance further. But we try to design a general approach here, and we found that this algorithm with the function in (8) performed very well from simulation results.

V. Simulation Results

In this section, we present and discuss simulation results to show the performance of the proposed algorithms.

In the simulation, WiFi signal sensing was considered to be our application. We selected three popular Android-based smart phones, Google Nexus S [15], Samsung I9000 and S5830 [19], as our sensing devices. The energy consumed by these phones for conducting a WiFi scan was measured by the Monsoon power monitor [12] (particularly designed for mobile phones), which are summarized in the following table. In the simulation, the mobile phone of a user was randomly selected from these three kinds of Android phones. The sensing range r was set to 7 meters. Since most of current mobile phone sensing projects were conducted in urban areas, we picked a typical urban area to carry out simulation runs. As shown in Fig. 2 obtained from the Google Map, the target region is located at Manhattan, NY, which spans 4 blocks from west to east with a total length of 1.135km and 4 blocks from south to north with a total width of 0.319 km, and includes the 6th,7th,8th Avenues and the 45th, 46th,47th Streets.

TABLE II
Energy Consumption of a WiFi Scan

Phone Models	Energy Consumption(μAh)
Google Nexus S	30.99
Samsung S5830	16.25
Samsung I9000	54.08

We used a mobility model similar to the well-known Manhattan model [2] to generate mobile users' moving trajectories. Specifically, each user was assumed to enter the target region from a road at a random time, randomly pick a speed from $\{2, 4\}$ meters per second, move towards an intersection, and then move straight ahead with a probability of 50% and turn left or right with equal probabilities (i.e., 25%). The trajectory of each user was constructed with evenly paced sample points (6 meters between two consecutive ones) and points on critical locations (such as intersections and road heads.) The location data were collected from the Google Map using its API.

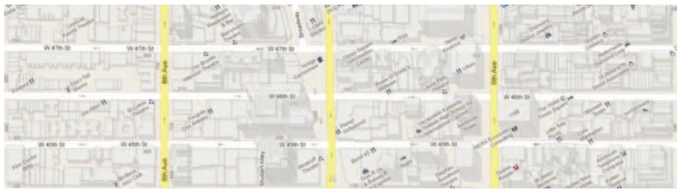


Fig. 2. The target region

We compared our algorithms, the prediction-based algorithm (labeled as "Prediction-Based"), the function-based algorithm (labeled as "Function-Based"), the optimal MECSS algorithm (labeled as "MinTotalEnergy"), the optimal FECSS algorithm (labeled as "FECSS") against a baseline approach in which every user performs a WiFi scan every 3 seconds. The total energy consumption, the variance of the number of sensing times, the maximum number of sensing times were used as performance metrics to show the energy consumption as well as fairness. In the simulation scenario, we increased the number of users from 25 to 50 with 5 as the step size. The simulation results are presented in Figs. 3–6.

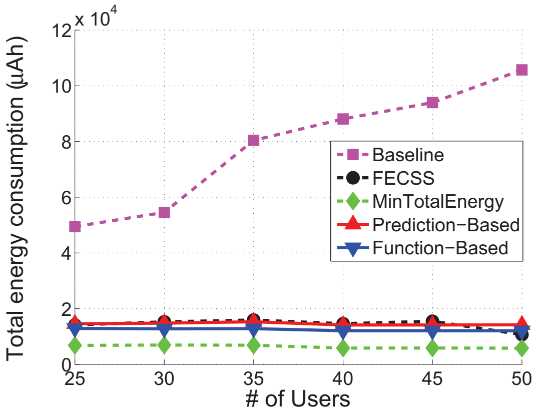


Fig. 3. The total energy consumption

From these simulation results, we can make the following observations.

1) From Fig. 3, we can see that in terms of total energy consumption, all the proposed algorithms perform very well. Specifically, compared to the baseline approach, the optimal MECSS algorithm, the optimal FECSS algorithm, the prediction-based algorithm and the function-based algorithm significantly reduce energy consumption by 91%, 80%, 80% and 82% respectively, on average. The optimal MECSS algorithm is certainly the best in terms of this metric. The optimal FECSS algorithm tries to minimize total energy consumption

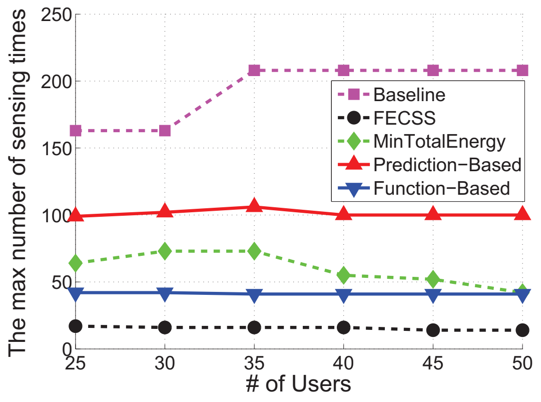


Fig. 4. The maximum number of sensing times

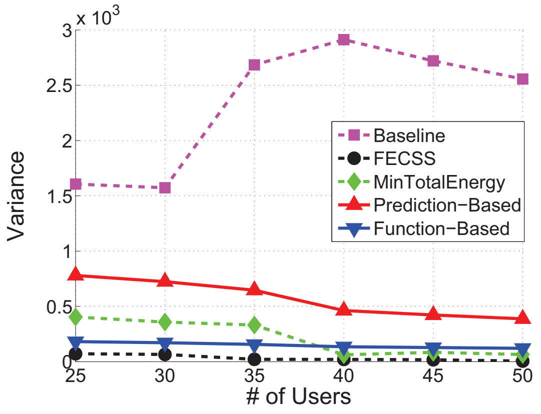


Fig. 5. The variance of the number of sensing times

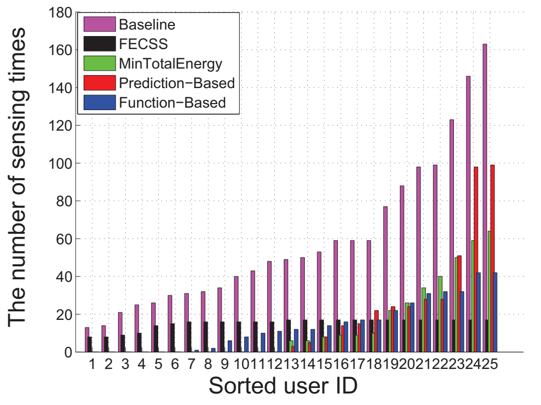


Fig. 6. The number of sensing times VS. the sorted user ID

under the constraint of achieving the min-max number of sensing times. Therefore, the total energy consumption given by this algorithm is larger than the minimum value. In addition, the function-based algorithm perform very well: close to the optimal MECSS algorithm and better than the prediction-based algorithm and the optimal FECSS algorithm (in terms of total energy consumption).

2) In Fig. 4, we show the fairness of the sensing schedules given by each algorithm in terms of the maximum number of sensing times. Since one of the constraints of the FECSS problem is to achieve the min-max number of sensing times. Therefore, we first used the maximum number of sensing times as a metric to evaluate the fairness performance. Clearly, the optimal FECSS algorithm is the best in terms of this metric since it is guaranteed to produce a solution in which the maximum number of sensing times is minimum among all possible solutions. This was verified by the results in Fig. 4. None of the other four algorithms can provide any guarantee for this metric. Not surprisingly, the baseline approach still performs worst. This is because the number of sensing times given by the baseline approach depends on how long a user stays in the target region, which can be arbitrarily large. An interesting observation is that the function-based algorithm outperforms both the optimal MECSS algorithm and the prediction-based algorithm.

3) In Fig. 5, we also used the variance of the number of sensing times as a metric to evaluate the fairness performance of algorithms. For all algorithms presented here, their performance in terms of variance matches that in terms of the maximum number of sensing times. Specifically, the optimal FECSS algorithm performs best as expected and the baseline approach is still the worst one. The function-based algorithm performs well too. On average, the optimal MECSS algorithm, the optimal FECSS algorithm, the prediction-based algorithm and the function-based algorithm outperform the baseline approach by 89%, 98%, 72% and 93% respectively.

In addition, we also present a bar graph in Fig. 6 to show how the number of sensing times is distributed over 25 users. In this figure, x-axis is the sorted user ID. As can be clearly seen, the number of user sensing times is distributed quite evenly over all the users if the optimal FECSS algorithm is used to determine the sensing schedule. If the baseline approach is used, every user needs to sense a few times, however, the distribution is not even at all.

4) From Fig. 3, we can also see that the total energy consumption given by a proposed algorithm does not increase (decreases slowly in most cases) with the number of users. As long as a sensing task is undertaken by mobile users in the target region collaboratively, more users usually offer more flexibility for sensing scheduling, which should better performance on energy consumption. As expected, the total energy consumption given by the baseline approach increases sharply with the number of users. This is because with this algorithm, each user senses individually without any collaborations. Hence, increasing the number of users does not necessarily bring any benefits. This observation well justifies the advantage of using collaborative sensing. Similar observations can be made for the other two metrics from Figs. 4–5.

In short, we can make two conclusions from the discussions above: 1) Compared to traditional mobile phone sensing without collaborations, collaborative sensing significantly reduces energy consumption. 2) The proposed function-based algorithm perform well in terms of both total energy consumption and fairness.

Since the function-based algorithm seems a promising method for collaborative sensing, we decided to study it further via simulation by investigating how the value of c (the tunable parameter in the exponential function) affects fairness. In this scenario, $N = 50$ and we used the same input as before. From the results in Figs. 7–8, we can see that the maximum number of sensing times and the variance given by the algorithm decrease with the value of c as expected. However, we found that if we increased it to a value larger than 2, we lost full coverage of some roads in the target region in some cases, which is not acceptable.

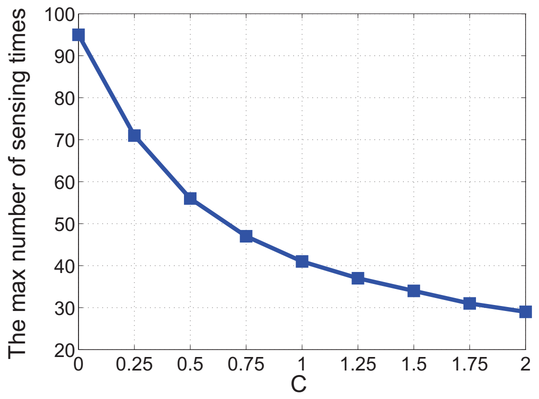


Fig. 7. The value of c VS. the maximum number of sensing times

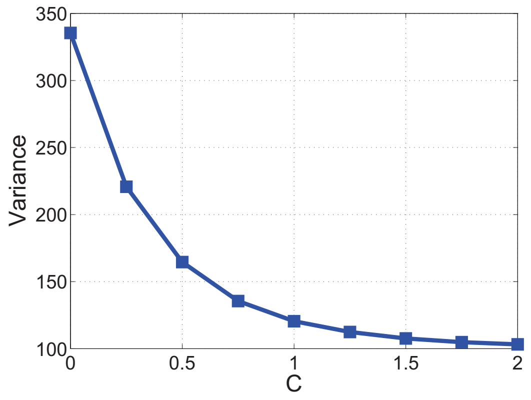


Fig. 8. The value of c VS. variance

VI. Conclusions

In this paper, we proposed to leverage cloud-assisted collaborative sensing to reduce energy consumption for mobile phone sensing applications. By assuming the moving trajectory of each mobile user is known in advance, we presented a polynomial-time algorithm to obtain minimum energy sensing schedules, which can be used to show potential energy savings that can be brought by collaborative sensing and can serve as a benchmark for performance evaluation. We also presented an algorithm to achieve a good tradeoff between total energy consumption and fairness. Under realistic assumptions, we presented two practical and effective heuristic algorithms: the prediction-based algorithm and the function-based algorithm. It has been shown by simulation results based on real energy consumption and location data that compared to traditional sensing without collaborations, collaborative sensing significantly reduces energy consumption, and the proposed function-based algorithm performs well in terms of both total energy consumption and fairness.

References

[1] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows (3rd Edition)*, John Wiley & Sons, 2005.

[2] F. Bai, N. Sadagopan, and A. Helmy. The IMPORTANT framework for analyzing the Impact of Mobility on Performance Of RouTing protocols for Ad hoc NeTworks, *Ad Hoc Networks*, Vol. 1, No. 4, pp. 383-403.

[3] S. Consolvo *et al.* , Activity sensing in the wild: a field trial of ubifit garden, *Proceedings of ACM Conference on Human Factors Computing Systems*, 2008, pp. 1797-1806.

[4] I. Constandache, R. Choudhury and I. Rhee, Towards mobile phone localization without war-driving, *Proceedings of IEEE Infocom'2010*.

[5] T. Dang, W. Feng and N. Bulusu, Zoom: A multi-resolution tasking framework for crowdsourced geo-spatial sensing, *Proceedings of IEEE Infocom'2011*, pp. 501–505.

[6] N. D. Lane *et al.* , A survey of mobile phone sensing, *IEEE Communications Magazine*, Vol. 48, No. 9, 2010, pp. 140–150.

[7] J. Lee and B. Hoh, Sell your experiences: a market mechanism based incentive for participatory sensing, *Proceedings of IEEE PerCom'2010*, pp. 60–68.

[8] K. Lin, A. Kansal, D. Lymberopoulos and F. Zhao, Energy-accuracy trade-off for continuous mobile device location, *Proceedings of IEEE MobiSys'2010*, pp. 285–297.

[9] H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury and A. Campbell, The Jigsaw continuous sensing engine for mobile phone applications, *Proceedings of ACM SenSys'2010*, pp. 71–84.

[10] S. Madhani, M. Tauil and T. Zhang, Collaborative sensing using uncontrolled mobile devices, *Proceedings of International Conference on Collaborative Computing, Networking, Applications and Worksharing*, 2005.

[11] E. Miluzzo *et al.* , Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application, *Proceedings of ACM SenSys'2008*, pp. 337–350.

[12] Monsoon Inc., *http://www.msoon.com/LabEquipment/PowerMonitor/*

[13] M. Mun *et al.* , PEIR, the personal environmental impact report, as a platform for participatory sensing systems research, *Proceedings of ACM MobiSys'2010*, pp. 55–68.

[14] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, A. Campbell, Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones, *Pervasive Computing, Lecture Notes in Computer Science*, 2010, pp. 355–372.

[15] Google Nexus S, *http://www.google.com/nexus/#*

[16] R. Rana, C. Chou, S. Kanhere, N. Bulusu and W. Hu, Ear-phone: an end-to-end participatory urban noise mapping system, *Proceedings of ACM/IEEE IPSN'10*, pp. 105-116.

[17] S. Reddy, MobiSense - mobile network services for coordinated participatory sensing, *Proceedings of IEEE ISADS'2009*.

[18] A. Saipulla *et al.* , Barrier coverage with sensors of limited mobility, *Proceedings of ACM MoboHoc'2010*, pp. 201–210.

[19] Samsung Android phones, *http://www.samsung.com/global/microsite/galaxys/*

[20] Sensorly, *http://www.sensorly.com*

[21] J. Tang, G. Xue and W. Zhang, Maximum throughput and fair bandwidth allocation in multi-channel wireless mesh networks, *Proceedings of IEEE Infocom'2006*.

[22] N. Thepvilojanapong, S. Konomi, Y. Tobe, Y. Ohta, M. Iwai, K. Sezak "Opportunistic collaboration in participatory sensing environments", *Proceedings of ACM MobiArch'2010*, pp. 39–44.

[23] H. Weinschrott, F. Durr and K. Rothermel, StreamShaper: coordination algorithms for participatory mobile urban sensing, *Proceedings of IEEE MASS'2010*, pp. 195–204.

[24] L. A. Wolsey, *Integer Programming*, John Wiley & Sons Inc., 1998.

[25] C. W. Yu, C. H. Wang, L. C. Hsu and K. J. Cheng, Coverage algorithms in GPS-less wireless mobile sensor networks, *Proceedings of ACM Mobility'2008*.

[26] S. Zhou, M-Y Wu and W. Shu, Finding optimal placements for mobile sensors: wireless sensor network topology adjustment, *Proceedings of IEEE Circuits and Systems Symposium on Emerging Technologies: Mobile and Wireless Communication*, 2004, pp. 529-532.