Enabling Green Mobile Crowd Sensing via Optimized Task Scheduling on Smartphones

Jing Wang, Jian Tang, Xiang Sheng, Guoliang Xue and Dejun Yang

***Abstract*—In a mobile crowd sensing system, a smartphone undertakes many different sensing tasks that demand data from various sensors. In this paper, we consider the problem of scheduling different sensing tasks assigned to a smartphone with the objective of minimizing sensing energy consumption while ensuring Quality of SenSing (QoSS). First, we consider a simple case in which each sensing task only requests data from a single sensor. We formally define the corresponding problem as the Minimum Energy Single-sensor task Scheduling (MESS) problem and present a polynomial-time optimal algorithm to solve it. Furthermore, we address a more general case in which some sensing tasks request multiple sensors to report their measurements simultaneously. We present an Integer Linear Programming (ILP) formulation as well as an effective polynomial-time heuristic algorithm, for the corresponding Minimum Energy Multi-sensor task Scheduling (MEMS) problem. Extensive simulation results show that the proposed algorithms achieve over 79% energy savings on average compared to a widely-used baseline approach, and moreover, the proposed heuristic algorithm produces close-to-optimal solutions.**

I. Introduction

Most smartphones are equipped with various powerful embedded sensors including microphone, camera, GPS, accelerometer, gyroscope, WiFi/3G/4G interfaces, etc. Moreover, wearable devices (i.e., Smart Watches, Fitbit, Sensordrone [12]) are booming, which can be connected to a smartphone Bluetooth interface to extend its sensing capability.

Pervasive mobile sensors can enable applications and services in various domains such as environmental monitoring, social networking, healthcare, transportation, safety, etc. In this paper, we consider a general-purpose mobile crowd sensing system, such as PRISM [3] and Medusa [9]. In this system, a service user can send a sensing request via a web portal, which will then be pushed to a set of smartphones. They will then undertake the corresponding sensing task by collecting data from one or multiple mobile sensors and returning them back to the user via a server. Unlike those systems targeting at specific applications (i.e., Ear-phone [11] for environment monitoring and SociableSense [10] for social networking), it is a multi-application multi-task system that supports a large variety of sensing applications. On one hand, a sensing task is dispatched to many smartphones for data collections; on the other hand, a smartphone undertakes many different sensing tasks that demand data from various sensors.

Collecting data from smartphone sensors is energy consuming. Specifically, for some sensors (such as WiFi interface), a scan needs to be actively performed to obtain a measurement; while for some other sensors (such as accelerometer) that are always working, a thread needs to be spawn to obtain its readings. Moreover, it has also been shown [5], [15], [19] that some sensors, including GPS, are power hungry. Hence, battery may be drained quickly if sensing activities are not carefully managed in an energy-efficient manner.

In this paper, we study sensing task scheduling on a smartphone with the objective of minimizing sensing energy consumption while guaranteeing Quality of SenSing (QoSS). Scant attention has been paid to the problem of determining how to schedule sensing tasks assigned to a smartphone subject to QoSS constraints, which is the main focus of this paper. Intuitively, we aim to save energy from the following two aspects: 1) Instead of periodically collecting sensor readings, we can strategically schedule sensor data collection activities for a given set of tasks to minimize energy consumption without violating their QoSS constraints. 2) Since multiple sensing tasks may request data from common sensors at the same or similar times, sensor data can be shared among them to avoid redundant efforts.

In this paper, we first consider a simple case in which each sensing task only requests data from a single sensor (*single-sensor tasks*). Single-sensor tasks are quite common, for example, many location-dependent applications may just request smartphones to report their locations. Furthermore, we address a more general case in which some sensing tasks request multiple sensors to report their measurements simultaneously (*multi-sensor tasks*). For example, some sophisticated applications may apply machine learning techniques to different sensor readings to infer non-trivial information from mobile users or their environment such as the mobile social networking application Sociablesense [10]. We summarize our contributions in the following:

- We formally define the problem of scheduling a set of single-sensor tasks as the Minimum Energy Single-sensor task Scheduling (MESS) problem and present a polynomial-time optimal algorithm for this problem.
- We present an Integer Linear Programming (ILP) formulation for the Minimum Energy Multi-sensor task Scheduling (MEMS) problem, and present an effective heuristic algorithm to solve it in polynomial time.
- We present extensive simulation results based on real data on sensor energy usages to show the proposed

Jing Wang, Jian Tang, Xiang Sheng are with Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244. Email: {jwang93, jtang02, xsheng}@syr.edu. Guoliang Xue is with Ira A. Fulton Schools of Engineering, Arizona State University, Tempe, Arizona 85287. Dejun Yang is with Department of Electrical Engineering and Computer Science, Colorado School of Mines, Golden, CO 80401. This research was supported in part by NSF grants 1217611, 1218203, 1444059 and 1421685. The information reported here does not reflect the position or the policy of the federal government.

978-1-4799-5952-5/15/$31.00 ©2015 IEEE

algorithms achieve significant energy savings, compared to a widely-used baseline approach, and moreover, the proposed heuristic algorithm produces close-to-optimal solutions.

To the best of our knowledge, we are the first to conduct a comprehensive study for sensing task scheduling on a smartphone in the context of general mobile crowd sensing and present provably-good and practically efficient solutions.

II. Related Work

Sensing task scheduling and optimization have been addressed in the context of both application-specific and general mobile phone/crowd sensing. In [15], the authors presented an Energy Efficient Mobile Sensing System (EEMSS), in which they power only a minimum set of sensors and use appropriate sensor duty cycles for energy savings. The paper [19] presented an adaptive location-sensing framework that significantly improves the energy efficiency of smartphones via substitution, suppression, piggybacking, and adaptation of applications location-sensing requests. In [5], Lin *et al.* studied energy-accuracy trade-off for continuous mobile device location, and designed and prototyped an adaptive location service for mobile devices, a-Loc, which helps reduce this battery drain. In [10], Rachuri *et al.* proposed SociableSense, a smart phone based platform provides an adaptive sampling mechanism as well as models to decide whether to perform computation of tasks, such as the execution of classification and inference algorithms, locally or remotely. In [16], the authors introduced mechanisms for automated mapping of urban areas, which provide a virtual sensor abstraction to applications. They also proposed spatial and temporal coverage metrics for measuring the quality of acquired data. In [13], Sheng *et al.* presented algorithms for energy-efficient sensing scheduling and showed that significant power savings can be achieved by collaborative sensing via simulations. Incentive mechanisms have been proposed in [18] to attract mobile users to participate in sensing activities. Another line of related research is energy-efficient resource allocation in wireless sensor networks [2], [4], [17].

We summarize the differences from these related works as follows: 1) Unlike research works targeting at energy-efficient sensing scheduling or optimization for specific applications, such as [5], [10], [15], [19], we aim to address task scheduling in general mobile crowd sensing systems. 2) Most related works on general mobile crowd sensing, such as [13], [16], [18], essentially studied the problem of determining how to assign tasks to a group of participating smartphones. We, however, consider the problem of scheduling different sensing tasks assigned to a smartphone, which is mathematically different from their problems. 3) The problems here are also mathematically different from energy-efficient resource allocation problems in wireless sensor networks [2], [4], [17].

III. System Model

As mentioned above, we consider a general multi-application multi-task mobile crowd sensing system and focus on a participating smartphone that is requested to undertake a set of sensing tasks involving various sensors. First, we consider a simple case in which each sensing task only involves a single sensor (i.e., single-sensor tasks). A single-sensor task is given by a 4-tuple $\mathbf{S}_k = (k, j_k, \mathbf{\Omega}_k, q_k)$, where k is the task ID (that may include information about the user initiating the request); j_k is the index of the sensor (that is requested to take measurements); $\mathbf{\Omega}_k = \{t_1, \cdots, t_{N_k}\}$ is a sequence of time instants at which the sensor readings are requested to be collected, which we call *sensing time sequence*; and q_k is the Quality of SenSing (QoSS) requirement (which will be explained later). Note that many applications may simply request a smartphone to collect sensor readings periodically, i.e., time instants in $\mathbf{\Omega}_k$ are evenly distributed in the time domain. However, our sensing model and algorithms are not restricted to this case, i.e., $t \in \mathbf{\Omega}_k$ could be any arbitrary time.

For the sake of energy saving, we argue that a sensor measurement may not need to be taken exactly at the requested time instant because readings of some sensors (such as light, temperature, etc) may change slowly over time, i.e., they can be collected at time instants that are slightly different from the requested ones. However, we need to make sure that QoSS is maintained at an acceptable level. If a smartphone is requested to collect a reading from a sensor at time instant t, but it does so at t' instead, then we say the accuracy of this sensing action is $p_A(t, t') \in [0, 1]$. Certainly, the closer t' is to t, the larger the value of $p_A(t, t')$, i.e., the more accurate. Here, we aim to propose a general model for QoSS so any method can be applied to estimate the sensing accuracy (as long as it has the property just mentioned above and it reflects reality). Similar as in [5], a possible solution is to use a bell-shaped function within a value range between 0 and 1:

$$p_A(t, t') = e^{-\frac{(t-t')^2}{2\sigma^2}}, \tag{1}$$

where different $\sigma's$ can be used to model different sensor readings. Here, smaller σ corresponds to sensor readings that change quickly over time (such as GPS), while larger σ corresponds to those that change slowly over time (such as temperature and light). In our simulation, we used this function to model the accuracy. The model can even be extended by considering the impact of other factors, such as the current motion state (walking, running, driving, etc) of the mobile user, on the value of σ.

Suppose that a sensing task $\mathbf{S}_k = (k, j_k, \mathbf{\Omega}_k, q_k)$ requests a smartphone to collect sensor readings according to a time sequence $\mathbf{\Omega}_k = \{t_1, \cdots, t_{N_k}\}$. The phone does so according to a sensing schedule $\mathbf{\Gamma} = \{t'_1, \cdots, t'_{N_{k'}}\}$. If $\forall t \in \mathbf{\Omega}_k$, $p_A(t, t') \geq q_k$ (where t' is the time instant in $\mathbf{\Gamma}$ that is closest to t, and q_k is the QoSS requirement of the sensing task), we say the sensing schedule $\mathbf{\Gamma}$ meets the *QoSS requirement of the sensing task* $\mathbf{S}_k$ and denote it by $p_A(\mathbf{\Omega}_k, \mathbf{\Gamma}) \geq q_k$. Note that since different applications may demand different sensing accuracies, q_k is defined to an application-specific parameter that varies with sensing tasks.

Without loss of generality, we discretize the time domain by evenly dividing a given sensing scheduling period into

intervals (with equal durations) with a sequence of time instants $\mathbf{\Psi} = \{t_1, \cdots, t_i, \cdots, t_N\}$ and assume that sensor readings can only be taken at those instants. The finer the granularity, the better the QoSS (i.e., accuracy), but the higher the computational complexity. The scheduling problem then becomes the problem of finding the "best" subset of such time instants. Now we are ready to define the scheduling problem for the single-sensor task case, which is referred to as the *Minimum Energy Single-sensor task Scheduling (MESS)* problem and is formally presented below:

Unknown decision variables:

- Scheduling variable $x_{ij} \in \{0, 1\}$: $x_{ij} = 1$ if it is scheduled to collect a reading from sensor j at t_i; $x_{ij} = 0$, otherwise.

MESS:

$$\min_{\mathbf{X}=\langle x_{ij}\rangle} \sum_{j=1}^{M} w_j (\sum_{i=1}^{N} x_{ij}) \tag{2}$$

Subject to:

$$p_A(\mathbf{\Omega}_k, \mathbf{X}) \geq q_k, \quad \forall k \in \{1, \cdots, K\}. \tag{3}$$

In this formulation, a set of K sensing tasks $\mathbf{S}_k$ are given as input, the output is the scheduling matrix $\mathbf{X} = \langle x_{ij}\rangle$, and w_j is the energy usage for taking a reading from sensor j. The objective (2) is to minimize the total sensing energy consumption. Without abusing notations, constrains (3) ensure that the sensing schedule $\mathbf{X} = \langle x_{ij}\rangle$ meets the QoSS requirement of each sensing task.

Next, we address a more general case in which some sensing tasks request a smartphone to collect readings from multiple sensors simultaneously, i.e., multi-sensor tasks. In this case, we still use a 4-tuple $\mathbf{S}_k = (k, \mathbf{J}_k, \mathbf{\Omega}_k, q_k)$ to denote a sensing task, where $\mathbf{J}_k$ is the set of indices of sensors from which the task requests data. Note that a unique constraint here is that if a multi-sensor task requests data from multiple sensors then these sensor readings must be collected at exactly the same time such that they can be used to generate some meaningful results. Similar to its counterpart in a single-sensor task, the sensing time sequence $\mathbf{\Omega}_k$ of a multi-sensor task, is a sequence of time instants at which sensor readings are requested to be collected. q_k is again the QoSS requirement of a multi-sensor task. Similarly, any functions or methods can be used in this case to model QoSS as long as they have the properties mentioned above. A feasible solution could still be a bell-shaped function (1), whose σ, however, needs to be properly chosen with consideration for multiple sensors. A conservative approach is to take the minimum one.

Note that data collected from a sensor j in a multi-sensor task may be used by a single-sensor task $\mathbf{S}_k = (k, j_k, \mathbf{\Omega}_k, q_k)$ to fulfill its QoSS requirement if $j = j_k$. However, usually data from a single-sensor task are not sufficient to fulfill the QoSS requirement of a multi-sensor task unless multiple single-sensor tasks (with the same set of sensors as that of the multi-sensor task) are scheduled to collect data simultaneously.

Suppose that we are given a multi-sensor task $\mathbf{S}_k$ and a *feasible* scheduling matrix $\mathbf{X}$ (to be feasible, it has to meet the unique multi-sensor scheduling constraint mentioned above). If we can still use $p_A(\mathbf{\Omega}_k, \mathbf{X}) \geq q_k$ to denote that a sensing schedule satisfies the QoSS requirement of task k (which could be a single-sensor task or a multi-sensor task) without abusing notations, then the *Minimum Energy Multi-sensor task Scheduling (MEMS)* problem can be formally defined in the same way as the MESS problem. We omit the formal definition due to similarity and space limitation. Even though they can be presented in the same way, the MESS problem is a special case of the MEMS problem, and the MEMS problem is much harder since the constraints $p_A(\mathbf{\Omega}_k, \mathbf{X}) \geq q_k, \forall k \in \{1, \cdots, K\}$ imply that if k is a multi-sensor task, then readings from multiple requested sensors must be collected at the same time, and sensor data can be shared among multi-sensor tasks and single-sensor tasks.

IV. Single-Sensor Task Scheduling

In this section, we present a polynomial-time optimal algorithm for the MESS problem defined above.

For the MESS problem, if two tasks request data from two different sensors, they obviously don't interfere with each other; therefore, they can be scheduled independently. The trouble makers are those tasks that request data from a common sensor, which need to be scheduled jointly. So we can apply a divide-and-conquer technique here by dividing given tasks into a collection of non-overlapping subsets of tasks on a sensor-by-sensor basis and solve a simpler problem of scheduling a subset of sensing tasks that request data from a common sensor (which we call *Simplified MESS (SMESS)* problem). Next, we show that we can pre-process a set of sensing tasks according to their QoSS requirements such that the SMESS problem can be formulated to an ILP problem with a nice property.

As mentioned above, the scheduling period is discretized to a sequence of time instants $\mathbf{\Psi} = \{t_1, \cdots, t_i, \cdots, t_N\}$. According to the QoSS model, for a given sensing task $\mathbf{S}_k = (k, j_k, \mathbf{\Omega}_k, q_k)$ and a time instant $t \in \mathbf{\Omega}_k$, we can identify an interval within the scheduling period (i.e., a subset of continuous time instants in $\mathbf{\Psi}$) such that at least one sensor reading needs to be collected within the interval to meet the QoSS requirement. Specifically, a time instant $t_i \in \mathbf{\Psi}$ is in the interval as long as $p_A(t, t_i) \geq q_k$. We use $l(t, q_k)$ and $u(t, q_k)$ to denote the indices of starting and ending time instants of the interval respectively. Note that usually $t_{l(t,q_k)} \in \mathbf{\Psi}$ and $t_{u(t,q_k)} \in \mathbf{\Psi}$ are distributed on the two sides of t. Then we can formulate the SMESS problem to an ILP problem, which is formally presented in the following.

Unknown decision variables:

- Scheduling variable $x_i \in \{0, 1\}$: $x_i = 1$ if it is scheduled to collect a reading at t_i; $x_i = 0$, otherwise.

ILP-SMESS($\mathbf{\Phi}_j$):

$$\min \sum_{i=1}^{N} x_i \tag{4}$$

Subject to:

$$\sum_{i=l(t,q_k)}^{u(t,q_k)} x_i \geq 1, \quad \forall S_k \in \mathbf{\Phi}_j, \forall t \in \mathbf{\Omega}_k. \tag{5}$$

In this formulation, the set $\mathbf{\Phi}_j$ of sensing tasks requesting data from sensor j is given as input and the output is the schedule given by $\langle x_i \rangle$. The objective (4) is to minimize the total energy consumption. Constraints (5) ensure that for each time instant requested by a task, t, the measurement is taken at least once during the period $[t_{l(t,q_k)}, t_{u(t,q_k)}]$, i.e., the QoSS requirement of each task is guaranteed to be satisfied. Now we are ready to present the proposed algorithm for the MESS problem.

Algorithm 1 The Optimal Algorithm for MESS

Input: K sensing tasks $\{\mathbf{S}_1, \cdots, \mathbf{S}_K\}$
Output: The scheduling matrix $\mathbf{X} = \langle x_{ij} \rangle$

1: $\mathbf{\Phi}_j := \emptyset$, $\forall j \in \{1, \cdots, M\}$;
2: $\mathbf{\Phi}_{j_k} := \mathbf{\Phi}_{j_k} + S_k$; $\forall k \in \{1, \cdots, K\}$;
3: $x_{ij} := 0$; $\forall i \in \{1, \cdots, N\}, \forall j \in \{1, \cdots, M\}$;
4: **for** ($j := 1$ **to** M) **do**
5: **if** ($\mathbf{\Phi}_j \neq \emptyset$) **then**
6: Solve the LP relaxation of ILP-SMSS($\mathbf{\Phi}_j$) to obtain the scheduling vector $\langle x_i^* \rangle$;
7: $\langle x_{ij} \rangle := \langle x_i^* \rangle$, $\forall i \in \{1, \cdots, N\}$;
8: **end if**
9: **end for**
10: **return** $\langle x_{ij} \rangle$;

The algorithm first divides the given set of sensing tasks into a collection of non-overlapping subsets $\mathbf{\Phi}_j$ according to the sensors requested by them. Then these sensing task subsets will be fed to the ILP-SMESS as input to calculate the sensor-specific schedules ($\langle x_i^* \rangle$), which are then combined to form the final solution, i.e., scheduling matrix $\langle x_{ij} \rangle$. Note that instead of solving ILP-SMESS, we solve its LP relaxation (denoted by LP-SMESS), which can be done in polynomial time. We will show that doing so always produces integer optimal solutions to ILP-SMESS and the proposed algorithm solves the MESS problem optimally in polynomial time. First, we show that a nice property of ILP-SMESS.

Definition 1 (Totally Unimodular (TUM) [8]): A square, integer matrix $\mathbf{B}$ is called UniModular (UM) if its determinant $det(\mathbf{B}) = \pm 1$. An integer matrix $\mathbf{A}$ is called Totally UniModular (TUM) if every square, nonsingular submatrix of $\mathbf{A}$ is UM.

Definition 2 (Consecutive-ones Property [6]): If $\mathbf{A}$ is (or can be permuted into) a $0-1$ matrix in which for every row, the 1s appear consecutively, then $\mathbf{A}$ is TUM.

Lemma 1: The constraint matrix of LP-SMESS in its standard form is TUM.

Proof: Let $\mathbf{A}_{\bar{N} \times N}$ denote the constraint matrix given by (5), which is in the canonical form. $\bar{N}$ is the total number of constraints in (5). According to Definition 2, $\mathbf{A}_{\bar{N} \times N}$ is TUM since in each row of $\mathbf{A}_{\bar{N} \times N}$, 1s appear consecutively. However, LP-SMESS has additional constraints, which are given below:

$$x_i \leq 1, \forall i \in \{1, \cdots, N\}. \tag{6}$$

By adding slack variables, we can transform the constraint matrix of LP-SMESS given by both (5) and (6) from the canonical form into the standard form [8], which is given as follows:

$$\mathbf{G} = \left(\begin{array}{c|c|c} \mathbf{A}_{\bar{N} \times N} & -\mathbf{I}_{\bar{N} \times \bar{N}} & \mathbf{0} \\ \hline \mathbf{I}_{N \times N} & \mathbf{0} & \mathbf{I}_{N \times N} \end{array} \right),$$

where $\mathbf{I}_{N \times N}$ and $\mathbf{I}_{\bar{N} \times \bar{N}}$ are identity matrices. Next, we need to show that $\mathbf{G}$ is TUM.

Adding to a TUM matrix with a row or column that is a unit vector will preserve the total unimodularity [1]. So, without losing the total unimodularity, we can add rows with unit vectors iteratively to $\mathbf{A}_{\bar{N} \times N}$, yielding a TUM matrix:

$$\left(\begin{array}{c} \mathbf{A}_{\bar{N} \times N} \\ \hline \mathbf{I}_{N \times N} \end{array} \right).$$

Furthermore, more columns with unit vectors will be added with the total modularity preserved, resulting in:

$$\left(\begin{array}{c|c|c} \mathbf{A}_{\bar{N} \times N} & \mathbf{I}_{\bar{N} \times \bar{N}} & \mathbf{0} \\ \hline \mathbf{I}_{N \times N} & \mathbf{0} & \mathbf{I}_{N \times N} \end{array} \right).$$

In addition, total unimodularity will be preserved by multiplying some columns or rows in a TUM matrix with -1 [1]. Clearly, if we multiply the columns in the middle part of the above matrix with -1, we can obtain $\mathbf{G}$, which is still TUM. This completes the proof. ■

Theorem 1: Algorithm 1 is a polynomial-time optimal algorithm for the MESS problem.

Proof: According to Lemma 1 and [8], solving the LP relaxation of ILP-SMESS always produces integer optimal solutions to ILP-SMESS. Moreover, because any two sensing tasks requesting data from two different sensors can be scheduled independently, combining solutions to a series of ILP-SMESS($\mathbf{\Phi}_j$) can yield an optimal solution to the MESS problem.

In addition, the pre-processing (lines 1–3) takes $O(MN + K) = O(N + K)$ time since M (i.e., the number of sensors on a smartphone) can be considered as a small constant. The LP solving (lines 4–9) takes $O(MT(\text{LP-SMESS}) = O(T(\text{LP-SMESS})$ time, where $T(\text{LP-SMESS})$ is the time for solving the LP-SMESS (i.e., the LP relaxation of ILP-SMESS). Since LP-SMESS includes N variables and at most $(K\hat{N}_{\max}) + N$ constraints (where $\hat{N}_{\max} = \max_{1 \leq k \leq K} |\mathbf{\Omega}_k|$). Hence, Algorithm 1 is a polynomial-time algorithm. This completes the proof. ■

V. Multi-Sensor Task Scheduling

In this section, we first show the MEMS problem can be formulated to an ILP problem, which can be used to provide optimal solutions. Then we present an effective heuristic algorithm to solve it.

Multi-sensor task scheduling shares some similarities with single-sensor task scheduling: for a given multi-sensor task $\mathbf{S}_k = (k, \mathbf{J}_k, \mathbf{\Omega}_k, q_k)$ and a time instant $t \in \mathbf{\Omega}_k$, an interval within the scheduling period can also be identified such that the QoSS requirement can be met if readings are collected from all the sensors in $\mathbf{J}_k$ at least once within this interval. Again, $l(t, q_k)$ and $u(t, q_k)$ denote the indices of starting and ending time instants of the interval respectively. However, multi-sensor task scheduling differs from single-sensor task scheduling in the sense that for a multi-sensor task, readings must be collected from all the sensors in $\mathbf{J}_k$ at exactly the same time. Moreover, for multi-sensor task scheduling, we cannot simply divide all sensing tasks into a collection of non-overlapping subsets of tasks according to their sensors since it is possible (not necessarily always) for two sensor tasks, say $\mathbf{S}_k$ and $\mathbf{S}_{k'}$, to share data if $\mathbf{J}_k \bigcap \mathbf{J}_{k'} \neq \emptyset$. We can easily come up with a Non-linear Integer Programming (NIP) formulation for the MEMS problem.

NIP-MEMS:

$$\min_{\mathbf{X}=\langle x_{ij}\rangle} \sum_{j=1}^{M} w_j (\sum_{i=1}^{N} x_{ij})$$

Subject to:

$$\sum_{i=l(t,q_k)}^{u(t,q_k)} (\prod_{j\in\mathbf{J}_k} x_{ij}) \geq 1, \quad \forall k \in \{1, \cdots, K\}, \forall t \in \mathbf{\Omega}_k. \quad (7)$$

The objective is again to minimize the total energy consumption. In constraints (7), the non-linear term $\prod_{j\in\mathbf{J}_k} x_{ij}$ takes a value of 1 if and only if x_{ij} is 1, $\forall j \in \mathbf{J}_k$ (i.e., all the sensors in $\mathbf{J}_k$ take measurements simultaneously). Therefore constraints (7) ensure that for each requested time instant $t \in \mathbf{\Omega}_k$ of a task k, readings are collected from all the requested sensors together at least once during the interval $[t_{l(t,q_k)}, t_{u(t,q_k)}]$, in other words, both the unique multi-sensor task scheduling constraints and the QoSS requirements are met. Even though we have a mathematical programming formulation for the MEMS problem, an NIP problem is notoriously hard to solve.

Next, we show we can transform NIP-MEMS to an equivalent ILP problem. The transformation is not trivial. By introducing $(u(t, q_k) - l(t, q_k) + 1)$ new binary variables for each time instant $t \in \mathbf{\Omega}_k$, $y_i^{kt} = \prod_{j\in\mathbf{J}_k} x_{ij}$ $\forall i \in \{l(t, q_k), \cdots, u(t, q_k)\}$, the non-linear terms in constraints (7) can be replaced by newly introduced variables. Furthermore, since x_{ij} takes binary values, we can establish the connections between the new variables and the scheduling variables by $y_i^{kt} = \min_{j\in\mathbf{J}_k} x_{ij}$; i.e., $y_i^{kt} \leq x_{ij}, \forall j \in \mathbf{J}_k$. By dosing so, we can ensure if $y_i^{kt} = 1$, $x_{ij} = 1, \forall j \in \mathbf{J}_k$. In this way, we linearize the non-linear constraints (7). Then we can transform NIP-MEMS to an equivalent ILP problem, ILP-MEMS, which is presented in the following:

ILP-MEMS:

$$\min_{\mathbf{X}=\angle x_{ij}\rangle} \sum_{j=1}^{M} w_j (\sum_{i=1}^{N} x_{ij})$$

Subject to:

$$\sum_{i=l(t,q_k)}^{u(t,q_k)} y_i^{kt} \geq 1, \qquad \forall k \in \{1, \cdots, K\}, \forall t \in \mathbf{\Omega}_k; \quad (8)$$

$$y_i^{kt} \leq x_{ij}, \qquad \forall k \in \{1, \cdots, K\}, \forall t \in \mathbf{\Omega}_k, \forall j \in \mathbf{J}_k, \forall i \in \{l(t, q_k), \cdots, u(t, q_k)\}. \quad (9)$$

Even though it is easier to solve ILP-MEMS than NIP-MEMS, it may still take exponentially long time for a large-size problem instance. Since we aim to to solve the MEMS problem in an online manner, we need to design fast polynomial-time algorithms. We come up with a heuristic algorithm based on ILP-MEMS. The basic idea is to solve the LP relaxation of ILP-MEMS (instead of solving ILP-MEMS directly) and then round non-integer values to integers. Then the problem boils down to how to round. After an extensive empirical study and theoretical analysis, we found that the constraint matrix of the LP relaxation of ILP-MEMS (denoted by LP-MEMS) is unfortunately not (but seems close to be) TUM. However, an interesting finding is for most problem instances, most of scheduling variables take integer values if solving LP-MEMS. Therefore, we come up with a simple LP rounding based algorithm, which is formally presented as Algorithm 2.

Algorithm 2 LP Rounding based Algorithm for MEMS

Input: K sensing tasks $\{\mathbf{S}_1, \cdots, \mathbf{S}_K\}$
Output: The scheduling matrix $\mathbf{X} = \langle x_{ij}\rangle$

```
Solve the LP relaxation of ILP-MEMS to obtain the scheduling matrix ⟨x*_ij⟩;
for (i := 1 to N) do
    for (j := 1 to M) do
        if (x*_ij ≠ 0 and x*_ij ≠ 1) then
            x_ij := 1;
        else
            x_ij := x*_ij;
        end if
    end for
end for
return ⟨x_ij⟩;
```

In this algorithm, we first solve the LP relaxation of ILP-MEMS, which can be done in polynomial time. However, we may end up with values that are not 0 or 1, but fractional between 0 and 1. Those values will be simply rounded to 1. In this way, the QoSS requirement of each task is guaranteed

to be satisfied. So the solution will certainly be feasible. This simple rounding algorithm works very well on average cases (since as mentioned above, most scheduling variables take integer values after solving LP-MEMS), which will be shown by simulation results. The algorithm is obviously a polynomial-time algorithm.

VI. Performance Evaluation

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

We implemented a widely used baseline approach (labelled as "Baseline") for performance comparisons. The baseline approach schedules sensors to collect readings exactly at the requested time instants given by $\mathbf{\Omega}_k$. For fair comparisons, if a common sensor is requested to collect its reading at some time by multiple tasks, the baseline method does it only once for all of them. For the MESS problem, we compared our optimal algorithm (labelled as "Opt-MESS") with the baseline method. For the MEMS problem, the proposed LP Rounding based algorithm (labelled as "LP-based") was compared against the baseline approach and the optimal solutions provided by solving ILP-MEMS ("Opt-MEMS").

In the simulation, energy consumption was used as the primary metric for performance evaluation. We considered 6 frequently used embbed sensors, including GPS, light sensor, accelerometer, gyroscope, WiFi and 3G. We used real data on power usages of these sensors obtained from the power profile of a Google Nexus 4 [7] smartphone and multiplied them by estimated durations to obtain energy usages, which are summarized in the following table. As mentioned above, we used a bell-shaped function to model QoSS. The values of σ were set to different values, ranging from 6 to 16 minutes.

TABLE I
Sensor Energy Usages

Sensor	Energy (mAs)
Accelerometer	5
GPS	400
Gyroscope	7
Light sensor	2
WiFi	100
3G	240

The duration of the scheduling period was set to 12 hours (say from 8AM to 8PM), which is evenly divided into 2 minute intervals. As described in Section III, we obtained a sequence $\mathbf{\Psi}$ of evenly-spaced time instants (within the scheduling period), at which sensor readings can be collected. The sensing tasks were randomly generated. Specifically, $\mathbf{\Omega}_k$ in each task k was set to a sequence of evenly-spaced time instants. For a single-sensor task, the sensor was selected randomly from 6 available sensors mentioned above. While for a multi-sensor task, the set of sensors were randomly selected from the following combinations {GPS, WiFi}, {GPS, 3G}, {GPS, light}, {GPS, WiFi, 3G} and {GPS, accelerometer, gyroscope}. The input of a MEMS problem instance included both multi-sensor tasks and single-sensor tasks.

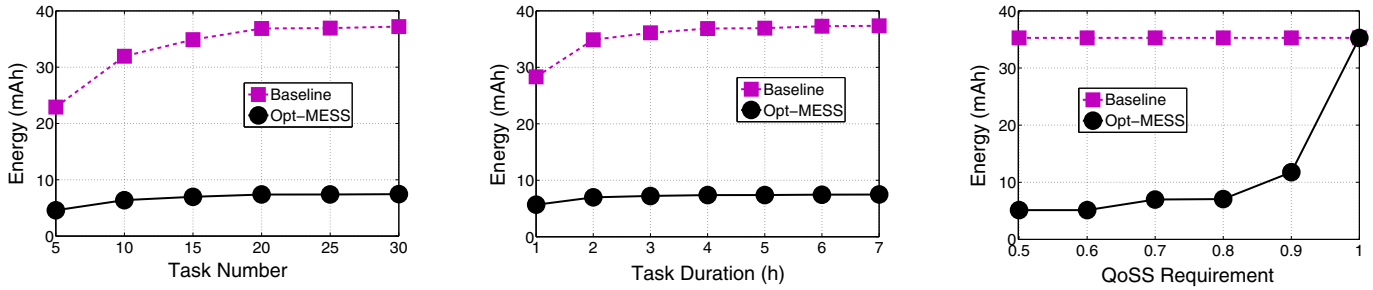
We evaluated the performance of the proposed algorithms extensively by varying the number of tasks, the duration of tasks, and the QoSS requirement of tasks in different simulation scenarios. Specifically, we came up with 6 scenarios in our simulation: the first three for MESS and the other three for MEMS. In scenario 1, the duration of each task was set to 2 hours with starting time randomly chosen from $[1, 10]$ such that their ending times do not exceed 12 (the length of the scheduling period). The QoSS requirement was fixed to 0.8. we changed the number of tasks from 5 to 30 with a step size of 5. In scenario 2, the number of tasks and the QoSS requirement were fixed to 15 and 0.8 respectively. We changed the duration of each task from 1 hour to 7 hours with a step size of 1 hour. Similar to scenario 1, the starting times were randomly generated in certain ranges accordingly such that their ending times do not exceed 12. In scenario 3, the starting times and durations of tasks were generated in the same way as scenario 1. The number of tasks was set to 15. We varied the QoSS requirement of each task from 0.5 to all the way to 1 with a step size of 0.1. The settings of scenarios 4, 5 and 6 were the same as those in scenarios 1, 2 and 3 respectively. But in these scenarios, we tested the algorithms for the MEMS problem instead of the MESS problem.

The simulation results for MESS and MEMS are presented in Fig. 1 and Fig. 2 respectively. Note that every number in these figures is the average over 50 runs. From these simulation results, we can make the following observations:

1) From Fig. 1, we can see that the optimal MESS algorithm reduces the sensing energy consumption significantly by 79.9% on average, compared to the baseline approach. Furthermore, energy savings become more and more significant when the input size (the number of tasks or the duration of tasks) become larger and larger. This leads us to believe that in a mobile crowd sensing system, significant energy savings can be achieved without sacrificing QoSS too much by strategically scheduling sensor data collections according to the requirements of tasks.

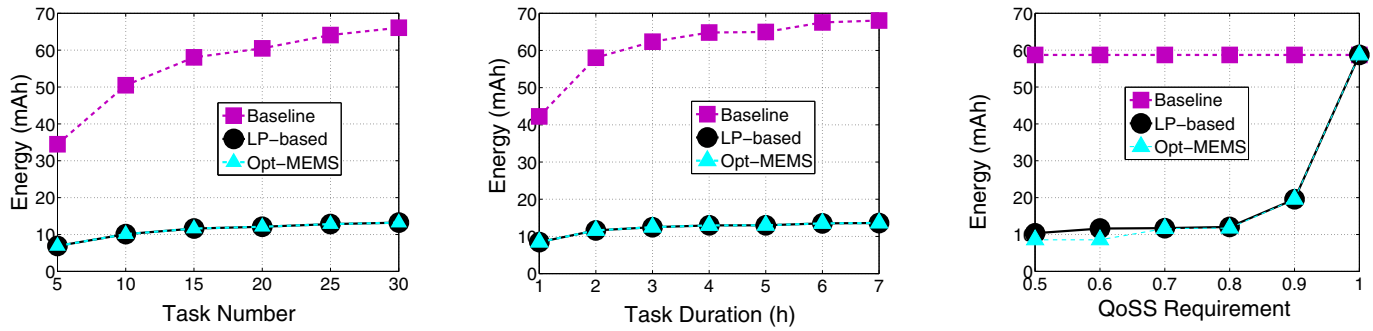
2) Very similar observations can be made for the multi-sensor task case. Specifically, from Fig. 2, we can see that the LP rounding based heuristic algorithm achieves substantial energy savings of 79.4% on average compared to the baseline. Moreover, the LP rounding algorithm almost always gives optimal solutions as expected.

3) An interesting observation can be made from Figs. 1(c) and 2(c) about the tradeoff between energy consumption and QoSS: no matter which algorithm is used, the energy consumption grows with the QoSS requirement monotonically. Increasing the QoSS requirement certainly increases energy consumption since more readings need to be collected at more time instants to fulfill the requirements. However, it turns out it is good to set QoSS requirements to relatively large values, say 0.8 or 0.9, since doing so does not lead to substantial increase on energy consumption. This finding provides a valuable insight about how to operate a mobile crowd sensing system in practice. When the QoSS requirement becomes 1 for all tasks, our algorithms perform exactly the



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 1. Performance of the MESS algorithms



(a) Scenario 4 (b) Scenario 5 (c) Scenario 6

Fig. 2. Performance of the MEMS algorithms

same as the baseline approach.

VII. Conclusions

In this paper, we considered the problem of scheduling sensing tasks assigned to a smartphone with the objective of minimizing sensing energy consumption while ensuring QoSS. First, we considered a simple case in which each sensing task only requests data from a single sensor and formulated the MESS problem correspondingly. We presented a polynomial-time optimal algorithm for this problem. Furthermore, we addressed a more general case in which some sensing tasks request multiple sensors to report their measurements simultaneously, and formulated the MEMS problem correspondingly. We presented an ILP formulation and an effective heuristic algorithm to solve it. Extensive simulation results showed that the proposed algorithms achieve over 79% energy savings on average compared to a widely-used baseline approach and the proposed heuristic algorithm for the MEMS problem produces close-to-optimal solutions.

References

[1] M. Bazaraa, J. Jarvis and H. Sherali, Linear programming and network flows, *John Wiley & Sons*, 2011.

[2] M. Cardei, M. T. Thai, Y. Li and W. Wu, Energy-efficient target coverage in wireless sensor networks, *IEEE Infocom'2015*, pp. 1976–1984.

[3] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee and A. Sharma, PRISM: platform for remote sensing using smartphones, *ACM MobiSys'2010*, pp. 63–76.

[4] H. Hassanein and J. Luo, Reliable energy aware routing in wireless sensor networks, *IEEE DSSNS'2006*, pp. 54–64.

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

[6] G. Nemhauser and L. Wolsey, Integer and combinatorial optimization, *John Wiley & Sons*, 1988.

[7] Google Nexus 4, *http://www.google.com/nexus/4*

[8] C. Papadimitriou and K. Steiglitz, Combinatorial optimization: algorithms and complexity, *Courier Dover Publications*, 1998.

[9] M-R Ra, B. Liu, T. L. Porta and R. Govindan, Medusa: a programming framework for crowd-sensing applications, *ACM MobiSys'2012*, pp. 337–350.

[10] K. Rachuri, C. Mascola, M. Musolesi and P. Rentfrow, SociableSense: exploring the trade-offs of adaptive sampling and computation offloading for social sensing, *ACM MobiCom'2011*, pp. 73–84.

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

[12] Sensordrone, *http://sensorcon.com/sensordrone/*

[13] X. Sheng, J. Tang and W. Zhang, Energy-efficient collaborative sensing with mobile phones, *IEEE Infocom'2012*, pp. 1916–1924.

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

[15] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishnamachari and N. Sadeh, A framework of energy efficient mobile sensing for automatic user state recognition, *ACM MobiSys'2009*, pp. 179–192.

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

[17] W. Ye, J. Heidemann and D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, *IEEE Infocom'2012*, pp. 1567–1576.

[18] D. Yang, G. Xue, X. Fang, and J. Tang, Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing, *ACM MobiCom'2012*, pp. 173–184.

[19] Z. Zhuang, K. Kim and J. Singh, Improving energy efficiency of location sensing on smartphones, *ACM MobiSys'2010*, pp. 315–330.