Enhancing Participant Selection through Caching in Mobile Crowd Sensing

Hanshang Li* Ting Li* Fan Li†‡ Weichao Wang§ Yu Wang*
*Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA
†Beijing Engineering Research Center of High Volume Language Information Processing
and Cloud Computing Applications, Beijing, 100081, China.
‡School of Computer Science, Beijing Institute of Technology, Beijing, 100081, China.
§Department of Software and Information Systems, University of North Carolina at Charlotte, Charlotte, NC 28223, USA

***Abstract*—With the rapid increasing of smart phones and their embedded sensing technologies, *mobile crowd sensing* (MCS) becomes an emerging sensing paradigm for performing large-scale sensing tasks. One of the key challenges of large-scale mobile crowd sensing systems is how to effectively select the minimum set of participants from the huge user pool to perform the tasks and achieve certain level of coverage. In this paper, we introduce a new MCS architecture which leverages the cached sensing data to fulfill partial sensing tasks in order to reduce the size of selected participant set. We present a newly designed participant selection algorithm with caching and evaluate it via extensive simulations with a real-world mobile dataset.**

I. Introduction

The widespread availability of smart phones equipped with built-in sensors has enabled a new sensing paradigm, *mobile crowd sensing* (MCS) [1], where tremendous data can be obtained and collected by the large group of selected mobile participants. It has been widely used in many applications, such as public safety [2], traffic planning [3]–[6], environment monitoring [7], [8], and urban dynamic mining [9], [10]. Compared with traditional static sensor networks, MCS does not require new infrastructures thus with low cost, while provides enriched spatio-temporal coverages and integrated human intelligences by leveraging the power of crowd. However, such large-scale sensing system also brings new challenges into the system design. Participant selection is one of them.

While the advantage of possible huge number participants enables massive mobile data sensing, how to effectively select a subset of participants to perform the desired sensing tasks become very challenging. Selecting larger number of participants can lead to better coverage of sensing tasks and better sensing quality, but also increases the cost of the overall sensing operation since performing sensing task is not free (costs energy of the smart phone). Therefore, how to pick the right amount and set of participants for certain tasks to minimize the sensing cost while guarantee certain coverage of tasks becomes the key issue.

Recently, there are several studies [11]–[16] beginning to address this important issue in MCS. Most of these methods formulate the participant selection problem as an optimization problem with certain constraints, and play tradeoffs among sensing cost, task coverage, energy efficiency, user privacy, and incentive. In this paper, we consider a dynamic participant selection problem with heterogeneous sensing tasks in a large-scale piggyback MCS system. Here, a piggyback MCS system [11], [12] allows the mobile user to upload the collected sensing data during its smartphone usage opportunities to save energy consumption for data transmission. Like the problem defined in [16], we assume that heterogeneous sensing tasks can arrive at any time and may have various temporal/spacial requirements. The goal of the participant selection problem is using minimum number of selected participants to perform the sensing tasks while still can guarantee certain level of probabilistic coverage.

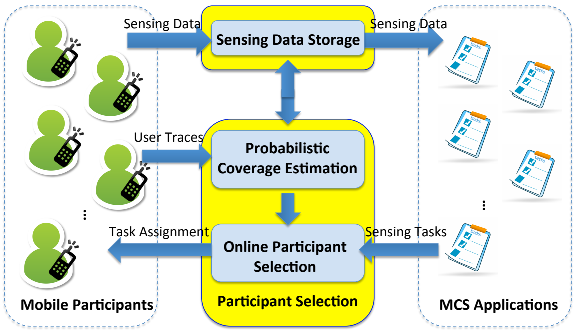


Fig. 1. The architecture of MCS system with caching.

One of the uniquenesses of this study is that we introduce a data storage component into the MCS system (as shown in Fig. 1) such that the sensing data can be cached to fulfill future incoming tasks. Hereafter, we also call this data storage the cache in our system. Caching mechanism has been widely used in many networking systems [17]–[19]. However, we believe that this is the first study of MCS with caching. With the newly introduced caching mechanism in MCS, the participant selection problem become more complex but with great potential to performance improvements. We then carefully design the new participant selection algorithms and corresponding caching strategies for such a system. In our design, we not only consider the knowledge obtained via historical call/location traces of mobile users but also the distribution of possible future tasks so that we can predict the future incoming tasks and estimate the contribution of particular participant to certain task set. Note that since we cannot foreknow when and where a participant will visit a place and make a phone call during the real crowding sensing period, our proposed online method try to estimate the coverage of current selected participants using the historical knowledge and dynamically adjust the selections.

We have conduct extensive simulations over a real-life mobile dataset (D4D data set [20]) to evaluate the proposed algorithm against existing solutions in different MCS settings. Our results show the proposed participant selection algorithm with caching can achieve stable task coverages while use much less number of participants against other solutions.

The rest of this paper is organized as follows. We first introduce our MCS system model and the participant selection problem in Section II. Then we present the detailed design of proposed participant selection algorithms with caches in Section III. Section IV presents our simulation results over a real-life mobile tracing dataset. Section V briefly reviews recent related works on participant selection in MCS. Finally, Section VI concludes this paper.

II. System Model and Participant Selection

A. System Model and Assumptions

As shown in Fig. 1, there are four main components in our mobile crowd sensing system: a large number of mobile participants, a set of crowd sensing applications, a participant selection mechanism, and a sensing data storage. The mobile participants are mobile users of smart devices who are willing to participant the sensing tasks. The crowd sensing applications are sensing information requesters which generate various sensing tasks continuously. The participant selection mechanism is the key of success of MCS system, in which sensing tasks from the MCS applications are assigned to particular sets of mobile participants. This has been the major focus of previous research in MCS. The data storage can temporarily caches sensing data collected and uploaded by selected participants, which later can be used to fulfill other sensing tasks. Note both participant selection and data storage components are established and operated by the mobile crowd sensing platform (they could be implemented in either centralized or distributed places).

The overall information flow in the system is as follows. Sensing tasks are generated by the applications and then sent to the participant selection mechanism. The selection of participants are made based on estimated coverages of participants to these tasks (based on historical user traces), and then the selected participants are assigned to perform these tasks. If the selected participants do occur at certain place and time, they will collect the sensing data and send it to the data storage and the corresponding applications. The data storage is responsible to keep or drop the collected data based on its caching strategy and size limit.

In this paper, we focus on the participant selection with the caches (data storage). We have the following assumptions.

1) Tasks: There is a set of m various mobile sensing tasks $S = \{s_1, \cdots, s_m\}$ generated by the crowd sensing applications within a particular time period T. Each task s_k can arrive at the system at any time, and the arriving time is defined as $t_{arrive}(s_k)$. The arriving time of incoming tasks subjects to certain probability distribution $\mathcal{P}$ in T (in the simulations, we use a Poisson distribution with a parameter λ). Each task s_k specifics a quadruple $< t_{begin}(s_k), t_{end}(s_k), l(s_k), freq(s_k) >$ as its target information in both temporal and spacial domains. The first two items define the beginning time and ending time of sensing task. $l(s_k)$ is the interested location of this task, and here we assume that there are r different sensing locations, denoted as $L = \{l_1, \cdots, l_r\}$. $freq(s_k)$ is the number of needed samples of this task at this location during the required time period. Note that here we assume that each sensing task only has one interested point in both temporal and spacial domains, but it is easy to relax such an assumption to handle complex sensing tasks with multiple interested points.

We further define two types of tasks. If $t_{arrive}(s_k) \leq t_{begin}(s_k)$, we call s_k a Type I task. This task acquires the sensing information in the future of its arriving. For such a task, we could assign participants to it after its arriving at the system. If $t_{arrive}(s_k) > t_{begin}(s_k)$, we call s_k a Type II task, which acquires the information in the past. For this type of tasks, we have to make assignments before the tasks arrive by prediction so that the collected information could meet the requirements of the tasks. For this type of tasks, the caching mechanism proposed in this paper becomes crucial.

2) Participants: There is a set of n mobile participants $P = \{p_1, \cdots, p_n\}$. Each participant p_i has his own visiting pattern or call pattern over both temporal and spacial domains. In this paper, we assume that the sensing tasks can be sent to the selected participants at any time by cellular service and the sensed data from the selected participants can only be updated to the system through piggyback [11], [12] during a phone call. Therefore, we are interested in the call patterns than the visiting patterns[1]. Each participant p_i has his own predicted probability $p(p_i, l_j, t)$ of making at least one phone call at time t and location l_j. This probability is a critical and necessary knowledge for participant selection. Since we cannot foreknow when and where a participant will place a phone call during the real crowding sensing period T (e.g., one week), we have to leverage knowledge from the historical traces. Here, we assume that for each user we have multiple rounds of call traces (e.g., K weeks), and each round of data denoted as D_i, $i = 1, \cdots, K$. Let $c_k(p_i, l_j, t)$ indicate whether p_i made one or more phone call at l_j and t in D_i (1 if it made, 0 otherwise). Then we simply estimate the call probability as follow,

$$p(p_i, l_j, t) = \frac{\sum_{k=1}^{K} c_k(p_i, l_j, t)}{K}.$$

Instead of this simple model, we can also consider more complex models, such as Bayesian/Markov model [15] or Poisson process [11], [12].

Based on this predicted information, the participant select mechanism can select a subset of participants to perform the

[1]In addition, the data set we used does not provide location information of each mobile phone user.

sensing task. Here we use an indicator $x(p_i, t)$ to represent whether user p_i is selected to participant for the task set. $x(p_i, t) = 1$ if user p_i is selected to participate at beginning time t, otherwise $x(p_i, t) = 0$. Here we assume a fixed sensing period τ for each selection. In other word, whenever a participant is selected to perform sensing tasks at a particular beginning time t, he will be active for a fixed time period τ. That means that this participant will perform sensing and upload data to data storage whenever he makes a phone call in the time period of $[t, t + \tau]$. Therefore, we have to restrict the selection of the same participant within τ as follows:

$$\sum_{t'=t}^{t+\tau} x(p_i, t') \leq 1 \textit{ for any } t \in [1, T] \textit{ and } i \in [1, n]. \quad (1)$$

Note that a single selected participant can perform the sensing task for multiple tasks and can also be selected multiple times at different time. The rewards to participant p_i are based on the number of his selections, i.e, $\sum_{t\in[1,T]} x(p_i, t)$.

3) Data Storage: The data storage is a data storage space with the total size of D, which can temporarily stores the sensing data uploaded by selected participants. The sensing data is formed by sensing data records. Each sensing data record r_i includes the time $r_i(t)$, location $r_i(l)$ and interested sensing information $r_i(d)$. Here, we assume that every single sensing data record has the same length, which means every of them needs the same size of storage space. Whenever selected participants place a phone call at the desired location and time, the sensing data is uploaded to the data storage. The data storage has the access to the full knowledge of tasks and assignments, and it can make decision on which sensed data should be cached based on certain caching strategy.

Based on participant selection for two types of tasks, there are also two types of collected sensing data uploaded to the data storage. For the first type, the sensing data could be utilized immediately by current tasks (Type I). Therefore, it will be forwarded directly to corresponding applications. In the same time, the storage will make its decision whether caches it for possible later tasks. For the second one, the sensing data are obtained based on prediction of future tasks (Type II). It will be cached in the storage for future usage and will not be forwarded to applications at current time.

Since the participant rewards are based on the number of their selections, a perfect situation is that the system stores all the data uploaded from selected participants for current or future utilization so that the number of selected participants can be reduced. However, such strategy will waste large number of storage space since most of the cached data may not be used for fulfill later tasks. Therefore, a smart caching strategy should be designed to determine whether to keep or drop data at any particular time.

B. Participant Selection Problem

Similar to [16], given the pool of candidates P and the crowd sensing tasks S, the participant selection problem aims to minimize total sensing cost while still satisfying certain level of probabilistic coverage of the tasks. The output of participant recruitment is a set of selected participants with selected time within the time cycle T, which showed by the indicator $\mathbf{x}(p_i, t)$ (hereafter, we use $\mathbf{x}(p_i, t)$ to represent the whole indicator set of all users and time). The overall optimization problem can be defined as:

$$\begin{aligned} &\min_x \sum_{i\in[1,n]} \sum_{t\in[1,T]} x(p_i, t) \\ &\textit{s.t.} \;\; C(\mathbf{x}(p_i, t)) \geq \gamma \textit{ and} \\ &\qquad \textit{Equation (1) on } \mathbf{x}(p_i, t). \end{aligned}$$

Here, the cost of sensing tasks is defined as the summation of all selections of participants $\sum_{i\in[1,n]} \sum_{t\in[1,T]} x(p_i, t)$. Once again that we assume a fix cost per selected participant for τ, such as energy cost of being active for τ. $C(\mathbf{x}(p_i, t))$ and γ are the overall expected coverage ratio of all tasks and the probabilistic coverage requirement, respectively.

For a particular task s_j, every time a single sensing data record r_k includes location $r_k(l) = l(s_j)$ and time $r_k(t)$ within time period $[t_{begin}(s_j), t_{end}(s_j)]$ is updated to the system, we consider that this task s_j is covered and accomplished by this record once. The accomplish frequency of each task could be accumulated by the number of different data records which could cover this task. Note that these sensing data records could be uploaded by single or multiple participants. Moreover, one particular participant could provide coverage to single or multiple tasks. Since the real coverage of tasks are based on the users actual calls, we can only use the call probability to estimate the expected probabilistic coverage of tasks. Let $C(p_i, s_j, t)$ equals to the number of the times that task s_j covered by user p_i who is selected starting from t. Then the coverage of task s_j can be defined as follows:

$$C(\mathbf{x}(p_i, t), s_j) = \min\Big(\sum_{t\in[1,T]} \sum_{i\in[1,n]} C(p_i, s_j, t)x(i, t), freq(s_j)\Big).$$

Note here if multiple selected users cover the same task, the coverage frequency cannot exceed $freq(s_j)$, i.e., fully covered. We can then define the overall coverage ratio of all tasks as follows:

$$C(\mathbf{x}(p_i, t)) = \frac{\sum_{j=1}^{m} C(\mathbf{x}(p_i, t), s_j)}{\sum_{j=1}^{m} freq(s_j)}.$$

The overall coverage constraint is not a full coverage requirement, instead a probabilistic coverage requirement (i.e., total task coverage is equal to or larger than a predefined coverage threshold γ).

Hereafter, we assume that the number of participant candidates are large enough so that if all of them are selected to participant then the sensing tasks can all be fulfilled. In other words, there always exists a feasible solution for this optimization problem. Such an assumption is reasonable for large-scale crowd sensing system. This participant selection problem can be proved NP-hard, by a simple reduction from minimum set cover problem (as proved in [16] for a simpler version of this problem). Therefore, in this paper, we are looking for efficient heuristics to solve it with the proposed caching storage. Though the participant selection problem defined so far is a static one, we actually want to solve it in an online version. In other words, the proposed participant selection algorithm is running with new tasks coming.

III. Participant Selection with Caching

In this section, we introduce our proposed participant selection algorithms and caching strategies. We first show how we predict the task coverage ratio $C(p_i, s_j, t)$ for any task based on the call probability obtained from historical data and also how we predict the future tasks.

A. Estimation of Coverage Ratio

To design our participant selection algorithm base on probability prediction, we need have an accurate estimation of the coverage ratio of each task by certain participants. Since each selected participant has independent probability to accomplish a task and each task is independent but may need multiple participants to accomplish, we need to estimate the coverage ratio $C(B, s_j, t)$ of task s_j by certain set of participant B at time t based on the call probability obtained from historical data. By doing so, we can have a simple greedy criteria in each round to select an individual user adding in the current selected participant set to maximize the increment of overall task coverage for all tasks. We can define the incremented task coverage for task s_j by adding p_i to current set of selected participants B as follows,

$$\Delta(B, p_i, s_j, t) = C(B + p_i, s_j, t) - C(B, s_j, t). \quad (2)$$

Then, the overall task coverage for all tasks by p_i in this round is

$$\Delta(B, p_i, t) = \sum_{j=1}^{m} \Delta(B, p_i, s_j, t). \quad (3)$$

To calculate $C(B, s_j, t)$, we need to estimate the probability that participants in B can fulfill task s_j, i.e., at least $freq(s_j)$ calls happened in the location of $l(s_j)$ and within the time period $[t_{begin}(s_j), t_{begin}(s_j) + \tau]$ from users in B. We first define the probability that x calls happened to fulfill task s_j as $C^x(B, s_j, t)$. Then

$$C(B, s_j, t) = 1 - \sum_{x=0}^{freq(s_j)-1} C^x(B, s_j, t). \quad (4)$$

Note that if a selected user contributes to s_j, he will make a call at $l(s_j)$ at time t and $t \in [t_{begin}(s_j), t_{begin}(s_j) + \tau]$ and a corresponding data sensing record r is updated. We call this event that the record r hits the task s_j. We have a call probability of such event $p(r) = p(p_i, l(s_j), t)$. For a particular task s_j, let R_j be all of the potential record hits s_j. To fulfill task s_j, we need at least $freq(s_j)$ records from R_j. To obtain $C^x(B, s_j, t)$, we can use the following formulation:

$$\sum_{\forall < r_1, \cdots, r_x > \in R_j^x} \prod_{r_i \in \{r_1, \cdots, r_x\}} p(r_i) \prod_{r_i \in R_j - \{r_1, \cdots, r_x\}} (1 - p(r_i)).$$

Here, $< r_1, \cdots, r_x > \in R_j^x$ is any x records can hit task s_j. Note that the maximal size of R_j is $n\tau$, while in reality it is much smaller. In addition, to further reduce the calculation cost, a dynamic programming can be used to obtain $C^x(B, s_j, t)$ from $C^{x-1}(B, s_j, t)$, which can be done in polynomial time.

When $freq(s_j) = 1$, Equation (4) can be simplified to

$$\begin{aligned} & C(B, s_j, t) \\ =& 1 - \sum_{p_i \in B, t_{begin}(s_j) \leq t' \leq (t_{begin}(s_j) + \tau)} (1 - p(p_i, l(s_j), t')). \end{aligned}$$

B. Prediction of Future Tasks

With Type II sensing tasks, we have to assign participants in advance to the coming of these tasks since they may request the sensing data in a particular time period before they come to the system. Recall that we assume that the coming task stream subjects to a Poisson distribution with parameter γ. Therefore, at particular time t, the number of future coming tasks $n(t)$ is given by

$$n(t) = \frac{(T - t)}{T} \lambda.$$

Each time a task s_j comes, it may request any combination of time period $t_{begin}(s_j)$ and location $l(s_j)$ as its target requirement with an independent probability $p(s_j, t_{begin}(s_j), l(s_j))$ (we call it task probability), which can be obtained from historical data of sensing requests[2]. Therefore, we have the overall probability of a task s_j will appear in the time period from current time t to T is calculated as:

$$p(s_j, t) = p(s_j, t_{begin}(s_j), l(s_j)) n(t). \quad (5)$$

To assign participants to the possible future tasks or estimate the coverage ratio of them by particular participant set, we basically modify Equation (3) to the following.

$$\Delta(B, p_i, t) = \sum_{s_j \in S_t} \Delta(B, p_i, s_j, t) + \sum_{s_j \notin S_t} p(s_j, t) \Delta(B, p_i, s_j, t). \quad (6)$$

Here, S_t represents the current task set which include all arrived tasks until t. Note that similar equations can also be defined by caching strategy to estimate the value of a record for current and future task sets.

C. Participant Selection Algorithm

As discussed above, we would like to design the participant selection algorithm as an online algorithm. First, the task streaming is dynamic, thus new tasks can come at any time within the time cycle. Second, the completion of tasks is dynamic, due to the mobility of users is dynamic. The coverage estimation above is based on the knowledge learned from historical data. However, the mobility pattern of users or distribution of tasks is random in real sensing period, thus the prediction may not be accurate and the coverage

[2]Since we do not have such traces, in our simulations, we assume that the probability distribution of $t_{begin}(s_j)$ is uniformly distributed while the one of $l(s_j)$ is proportional to the population nearby.

Algorithm 1 Online Algorithm for Participant Selection at Time t'

```
Input: participant pool $P$, call probability $p(p_i, l_j, t)$ for
    each user in $P$, task probability $p(s_j, t_{begin}(s_j), l(s_j))$,
    previous selected participant set $B_{t'-1}$, and current task
    set $S_{t'}$ (including tasks arrived at time $t'$).
Output: current selection $\mathbf{x}(p_i, t)$
1: update the current task set $B_{t'-1}$ and previous selection
   $\mathbf{x}(p_i, t)$ if there are new sensing data uploaded at data
   storage and partially fulfilling certain tasks from last time.
2: copy all previous selection $\mathbf{x}(p_i, t)$ from $B_{t'-1}$.
3: while $C(\mathbf{x}(p_i, t)) < \gamma$ based on $S_{t'}$ do
4:     for all $p_i \in P$ and $t \in [t', T]$ and $x(p_i, t) = 0$ do
5:         Calculate the improvement $\Delta(B_{t'-1}, p_i, t')$ by
           adding $p_i$ with starting time $t$, i.e., $x(p_i, t) = 1$, based
           on Equation (6))
6:     end for
7:     Select the user $p_i$ who leads to the largest coverage
       improvement, and set $x(p_i, t) = 1$
8: end while
9: return $\mathbf{x}(p_i, t)$
```

estimation may not reflect the true coverage. Therefore, in our online algorithm, we dynamically take new coming tasks into account and add more participants for unfulfilled sensing tasks whenever the overall estimated coverage can not meet the coverage requirement during the time cycle. On the other hand, if certain tasks are fulfilled and partially fulfilled when certain sensing data is updated at data storage, they will be removed or updated in current task set S_t and certain previously selected users can be withdrawn from the selected participant set B_t. The details of online algorithm is described in Algorithm 1. In each round, the algorithm basically repeatedly adding new participant which leads to largest coverage gain at current time until the estimated overall coverage reaches the requirement threshold.

D. Caching Operation and Strategies

So far, it seems that we did not discuss the cache operation yet. But actually caching has been used in Algorithm 1. First, in Line 1, if any sensing data is uploaded at data storage, it may triggers updates of task set and selected participants. If the record hits a particular task, the required frequency of that task will be reduced by 1. If frequency becomes zero (i.e, the task is fulfilled), the task will be removed from the task set. If certain selected participants cannot contribute to the updated task set, they can be removed from the current arrangements too. In addition, when we estimate the coverage improvement $\Delta(B_{t'-1}, p_i, t')$ in Line 5, we do consider the cached data from previous selected users. There are two scenarios of caching can be implemented there: *passive caching* and *active caching*. In the passive caching, the participant selection algorithm only assigns tasks that already arrived to the candidates. That means that the caching sensing data is only from the assigned participants for past tasks. In other words, the only difference between caching and no caching is whether the collected sensing data can be reused by other tasks. In this case, $\Delta(B_{t'-1}, p_i, t')$ can be estimated using Equation (3). In the active caching, the participant selection algorithm will assign and withdraw tasks dynamically during the whole time cycle. In addition, the assignments are not only based on tasks that already arrived but also the predicted future coming tasks. In this case, $\Delta(B_{t'-1}, p_i, t')$ can be estimated using Equation (6), in which both existing tasks and future coming tasks are considered. Note that for future coming tasks, the estimation is based on $p(s_j, t_{begin}(s_j), l(s_j)$ and $p(s_j, t')$. In the same time, such operation may add many unnecessary participants, thus we also allow the assignments can be withdrew when the corresponding tasks have been fulfilled. The principle is that each assignment can be withdrew with no cost at a particular time if and only if that time is before the begin time of that assignment. It can not be withdrew once an assignment starts, in other words, the selected participant in that assignment begins to upload sensing tasks whenever he makes a call.

In addition, there are two cases depending on the size of data storage in the proposed MCS system. If the cache space is infinite (i.e., $D = \infty$), we call it *infinite cache*. For this case, you may just want to cache every sensing data you received. If the cache space is limited by a finite number, we call it *finite cache*, where carefully caching strategy is needed when the space is full during a sensing data uploading. We consider three different strategies. The simplest is random cache. In this strategy, whenever the data storage is full, the system will randomly choose one record to be replaced by the next coming sensing date record. The second strategy is first in first out (FIFO). the oldest data record will be dropped when the cache is full. The third one is coverage based cache, in which we estimate the contribution of coverage of each record. The system will always drop the data record with least coverage ratio. We will test all of these three strategies in our simulations.

IV. Simulations

In this section, we conduct extensive simulations over a real-life mobile traces (D4D data set [20]) to evaluate the effectiveness of our proposed participant algorithms under different scenarios.

A. D4D Dataset and Sensing Task Generation

To simulate the large scale mobile crowd sensing, we choose a real life wireless tracing data from the cellular operator Orange for the *Data for Development (D4D) challenge*. The released D4D datasets [20] are based on anonymized Call Detail Records (CDR) of phone calls and SMS exchanges between $50,000$ Orange mobile users in Ivory Coast for about 20 weeks. We use the dataset of individual trajectories with high spatial resolution (*SET2* in D4D datasets), which contains 10 groups of the access records of antenna (cellular tower) of each mobile user. Each group of records are collected over a two-week period. But unfortunately, in each group of records, the user IDs were renumbered and anonymized, which makes

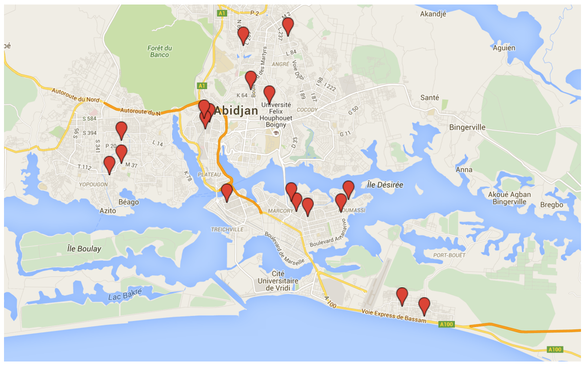


Fig. 2. Locations of cellular towers in Abidjan used as sensing locations.

impossible to merge them together. Thus, all of our MCS experiments are performed within a one week period (i.e., T is one week). We treat one hour as the smallest time unit, $T = 7 \times 24$. We perform simulations over five different weeks. We use the sequences of visited cellular towers of all users within these weeks to generate the call probability of each mobile user and location (i.e., cellular tower). We assume that the mobile users with the same user IDs are same users in all of these weekly call records. We choose a random set of users as candidate participants and a subset of 18 cellular towers as locations in MCS.

All the selected cellular towers are located in the region of Abidjan, the economic and former capital of Ivory Coast and the largest city in the nation. Fig. 2 shows the locations of these towers on the map of Abidjan. The area of Abidjan is informally composed of two parts (northern Abidjan and southern Abidjan) with ten formal boroughs, or communes, each being run by a mayor. One of them is covered by forest thus mobile call activities in that commune are much fewer to the other ones. Therefore, we choose to pick the cellular towers from the other nine communes. We have choose two towers from each communes with the most and second most number of mobile calls during a two weeks period. Thus, we have 18 cellular towers in total.

For each sensing task s_i, we need to pick its arriving time $t_{arrive}(s_i)$, location $l(s_i)$, starting time $t_{begin}(s_i)$, ending time $t_{end}(s_i)$, and $freq(s_i)$. $t_{arrive}(s_i)$ is generated by a Poisson distribution with parameter γ, while $t_{begin}(s_i)$ is randomly picked within 1 to T and $t_{end}(s_i)$ is fixed at τ (set to 24 hours). In other words, the duration of sensing period of a task is limited to one day. For location $l(s_i)$, it is chosen from 18 cellular towers based on the publicized population within the communes where the towers sit. In other words, the area with higher population has more chance to be chosen as the sensing target. For the frequency requirement, we simply set $freq(s_i) = 1$ for all tasks. Note that a task with $freq(s_i) = k$ can be approximated by k identical tasks with $freq(s_i) = 1$. For candidate participants, we randomly choose them from the mobile users with the highest number of times of visiting these towers. All parameters used are given in Table I.

In all simulations, we randomly generate MCS tasks based

TABLE I
PARAMETERS USED IN SIMULATIONS

Parameter	Value or Range
Unit of time	1 hour
Task life time τ	24 hours
Number of locations (towers) r	18
Number of tasks m or λ	$100, 150, 200, 250, 300$
Number of candidate participants n	$100, 200, 300, 400, 500$
Length of whole sensing cycle T	one week $= 7 \times 24$ hours
Total simulation period	Dec 5 2011 to Jan 8 2012
Coverage threshold γ	$0.4, 0.45, 0.5, 0.55, 0.6$

on the method discussed above, and apply different participant selection algorithms to select participants for all tasks. The selected participant will upload the sensing data around the location where he make calls during the assigned time interval (24 hours from the starting time). Based on the real traces, we evaluate how many tasks can be fulfilled with the selected participants. Here, a task is completed if and only if there is at least required number of calls made in the period of the lifetime of the task within the target location from the selected participants. Since the prediction of making a call is based on historical data, it is not possible to guarantee full coverage of all tasks or even the required portion of all tasks.

B. Tested Algorithms and Scenarios

Beside the proposed method, we also implement two simple algorithms and the one in [16] for comparisons. Most of them are greedy algorithms, where in each round a user is selected as the next participant of the MCS. Here are the four participant selection algorithms.

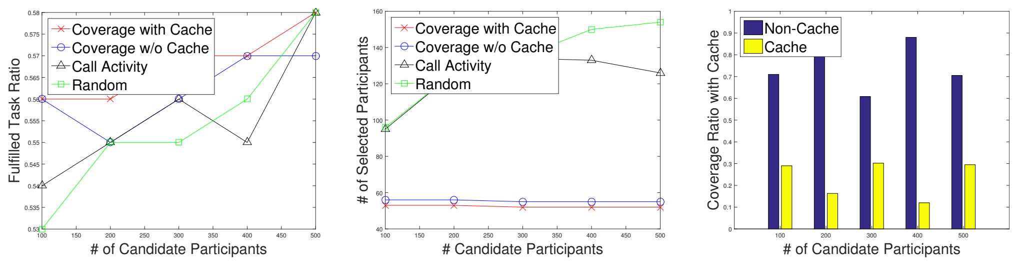
- Random: In each round, a random user is selected as the next participant of the MCS.
- Call Activity: In each round, the user with highest call activity is selected as the next participant.
- Coverage without Caching: In each round, the user with largest coverage improvement without caching is selected as the next participant [16].
- Coverage with Caching: In each round, the user with largest coverage improvement with caching is selected as the next participant.

In all experiments, we compare each algorithm using the following three measurement metrics.

- Number of selected participants: the number of selected participants[3] generated by the algorithm for the whole task set over the sensing period.
- Fulfilled task ratios: the ratio between the number of sensing tasks which are successfully performed by selected participants from the algorithm during the sensing period and the total tasks.
- Coverage ratio with caching: the portion of fulfilled tasks which are fulfilled by cached sensing data.

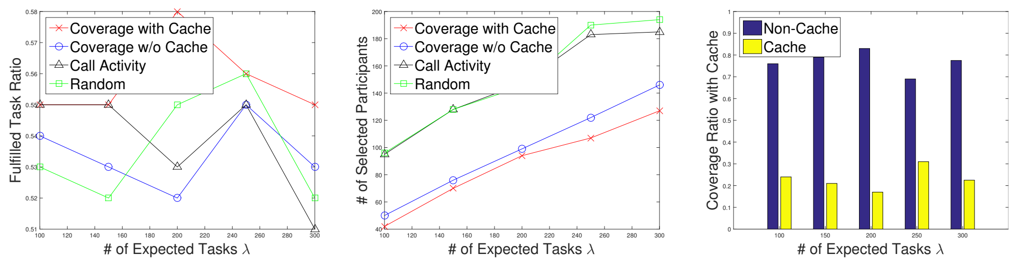
All results reported here are the average from multiple runs over different periods from the D4D data set.

[3]Note that a single user can be selected for multiple sensing periods (each of them lasts τ, e.g. $x(p_i, t_1) = 1$ and $x(p_i, t_2) = 1$) and that is counted as multiple participants.



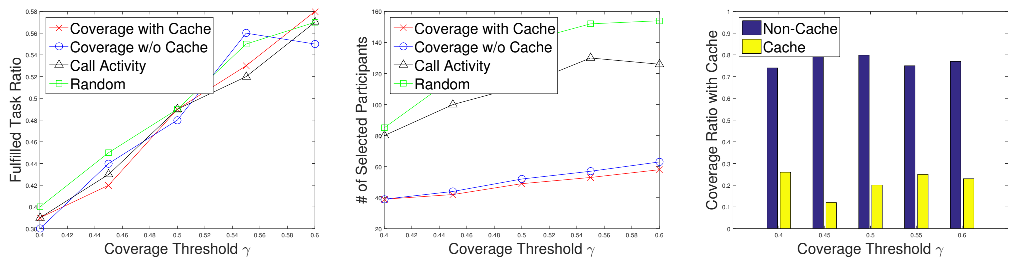
(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Fig. 3. Performance under Scenario A, when $\lambda = 200$, $\gamma = 0.6$ and $n = 100$ to 500.



(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Fig. 4. Performance under Scenario A, when $n = 300$, $\gamma = 0.6$ and $\lambda = 100$ to 300.



(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Fig. 5. Performance under Scenario A, when $n = 300$, $\lambda = 200$ and $\gamma = 0.4$ to 0.6.

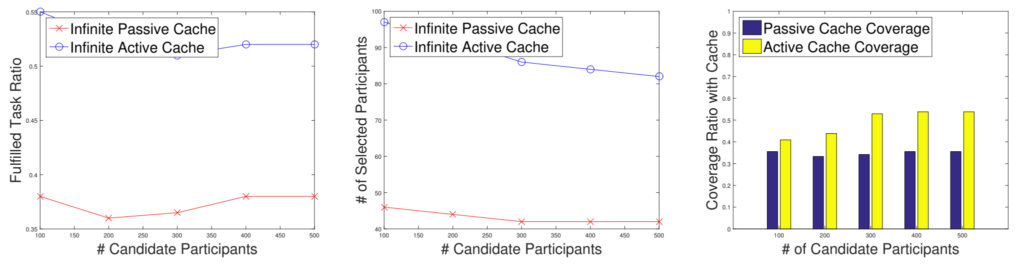
Moreover, we construct two simulation scenarios. In Scenario A, the task cycle type is single-cycle (e.g., one week) and all generated tasks are Type I. In Scenario B, the task cycle type is still single-cycle, tasks could be either Type I or II.

C. Performance under Scenario A

We first test the proposed algorithms (Coverage with Cache) against three existing solutions (Random, Call Activity, Coverage without Cache) when all sensing tasks are Type I. For this scenario, we consider infinite and passive cache.

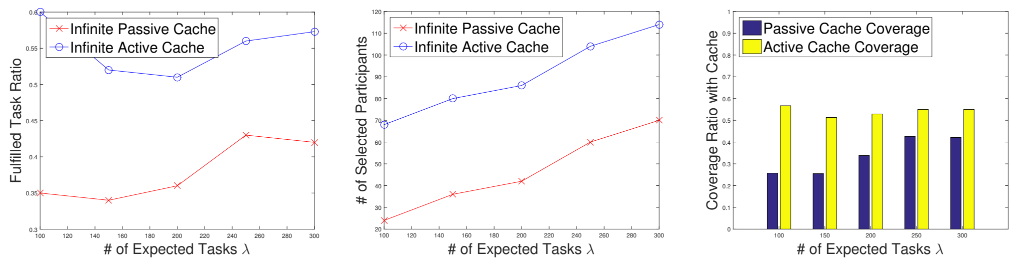
In the first set of simulations, we fix the parameter λ of the coming task stream to 200 and coverage threshold γ to 0.6, while varying the number of candidate participant from 100 to 500. Fig. 3 shows the performance comparison of four different algorithms. Fig. 3(a) shows that all methods have similar fulfilled task ratios, since all of them will keep add participants until the expected fulfilled ratio reaches the requirement. With more candidate participants, all of them can achieve better fulfilled task ratio since you have more choices. However, as shown in Fig. 3(b), the coverage based solutions have much fewer selected participants than the other two solutions and they are also stable with the increase of number of candidate participants. This shows the advantage of prediction of coverage in participant selection. In addition, the coverage-based algorithm with caching could select fewer participants than the one without caching. That is because when a task comes to the system, it may already be covered by some selected participant via caching. Thus the system may not assign or assign fewer participants to this task. Note the advantage of caching is not significant here, but it is mainly due to the sparseness of the D4D dataset. Fig. 3(c) shows the portion of fulfilled tasks by either cached records from previous selected participants or by newly assigned participants in the method of Coverage with Cache. The caching data contributed to 10% to 30% coverage.

In the second set of simulations, we fix the number of candidate participant at 300 and coverage threshold γ to 0.6, while varying the parameter λ of the coming task stream from 100 to 300. Fig. 4 shows the performance comparison of four different algorithms. Clearly, the number of tasks also affects the results. More tasks need more selected participants to fulfill. The four algorithms still have similar fulfilled task ratios but the method of Coverage with Cache uses the minimum



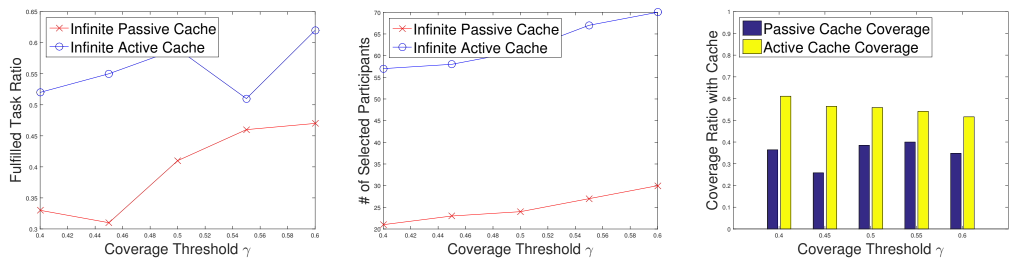
(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Fig. 6. Performance in Scenario B when $\lambda = 200$, $\gamma = 0.6$ and $n = 100$ to 500.



(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Fig. 7. Performance in Scenario B when $n = 300$, $\gamma = 0.6$ and $\lambda = 100$ to 300.



(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Fig. 8. Performance in Scenario B when $n = 300$, $\lambda = 200$ and $\gamma = 0.4$ to 0.6.

number of participants.

Last, we also test different values of the coverage threshold γ (as shown in Fig. 5). Both the fulfilled task ratio and the number of selected participants increase as the the threshold increases. In other words, high coverage requirements lead to higher fulfilled task ratios with larger selected participant sets.

D. *Performance under Scenario B*

Next we consider Scenario B with infinite cache, where tasks could be either Type I or Type II. We perform the same three sets of simulations as we did for Scenario A. We test both active caching and passive caching. Fig.s 6 to 8 are the results for the three sets of simulations, respectively. From these results, we can draw the following conclusions.

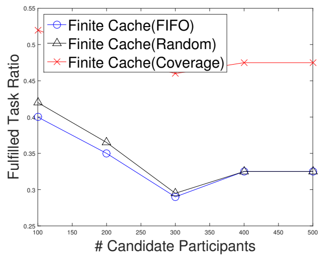
(1) The fulfilled task ratio of Active Caching is about 60 percent more than that of Passive Caching (Fig.s 6(a) to 8(a)). Recall that in Scenario B there are Type II sensing tasks which cannot be fulfilled by the passive caching. But the passive caching can dynamically assign participants to future coming tasks which leads to higher fulfilled ratios. But the cost of such advantage is more participants selected to perform the sensing tasks (Fig.s 6(b) to 8(b)).

(2) Similar to Scenario A, as the number of task or the coverage threshold increases, the number of selected participant increases (Fig. 7(b) and Fig. 8(b)). In addition, the number of selected participants decreases as the number of candidate participants increases (Fig.s 6(b)).

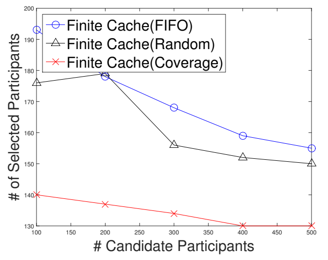
(3) The fulfilled tasks contributed by active caching could be about $40 \sim 60\%$ of the total fulfilled tasks (Fig.s 6(c) to 8(c)). This again shows the advantage of active caching due to Type II tasks.

E. *Performance with Different Caching Strategies*

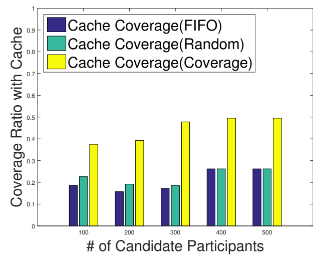
Now we test different caching strategies when the cache storage has limited space (finite cache). Three different caching strategies are implemented: random, FIFO, and coverage-based. We set the size of data storage $D = 500$, i.e., at most 500 records can be cached in the storage at any time. Fig.s 9 to 11 show the performance of these three strategies. It is obvious that the strategy based on coverage can achieve the best fulfilled task ratios with fewest selected participants. Also in term of the coverage ratios with caching, the hitting rate of coverage based method is as twice as much that of the



(a) fulfilled task ratio

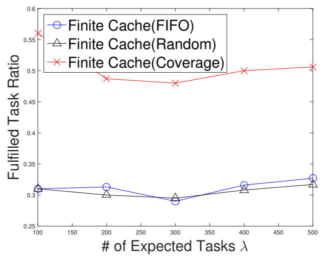


(b) # of selected participants

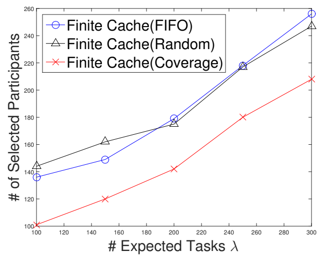


(c) coverage ratio with cache

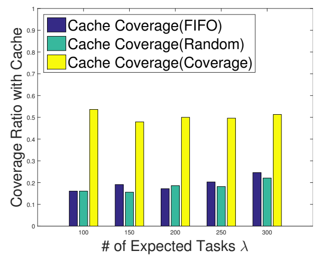
Fig. 9. Performance of different caching strategies when $\lambda = 200$, $\gamma = 0.5$ and $n = 100$ to 500.



(a) fulfilled task ratio

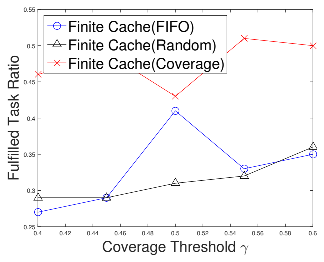


(b) # of selected participants

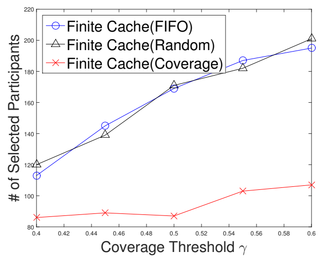


(c) coverage ratio with cache

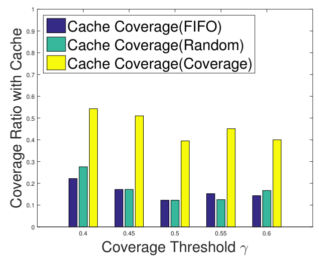
Fig. 10. Performance of different caching strategies when $n = 300$, $\gamma = 0.5$ and $\lambda = 100$ to 300.



(a) fulfilled task ratio



(b) # of selected participants



(c) coverage ratio with cache

Fig. 11. Performance of different caching strategies when $n = 300$, $\lambda = 200$, $\gamma = 0.4$ to 0.6 and $D = 500$.

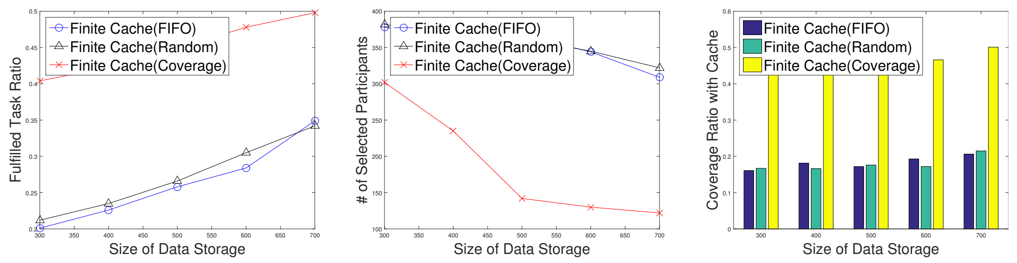
other two caching strategies. Clearly, it still effective to use estimated coverage as the metric in caching strategy.

F. Performance with Different Caching Sizes

In the last set of simulations, we vary the size of sensing data space D while fix the other parameters to evaluate the affection of caching size. Fig. 12(a) shows that the total fulfilled task ratio increases when the size of the data storage space increases. Moreover, the increasing of cache space size leads to fewer participants selected (in Fig. 12(b)). The reason is that more space for sensing records means more chance for effective records being utilized. Fig. 12(c) shows that the hitting rates of different caching schemes are relevantly stable.

V. Related Work

Mobile crowd sensing (MCS) has been widely used for different sensing applications [1]. To handle participant selection in large-scale MCS, different algorithms and systems has been proposed recently. For example, Zhang *et al.* [13] study offline participant selection in piggyback MCS for probabilistic coverage, which aims to select minimum number of participants to guarantee that the selected participants will make enough number of calls at certain percentage of the target locations over a fixed sensing period. The coverage requirement is different with our model. Xiong *et al.* [14] have investigated how to assure the asymptotically full coverage over a 13 tower region with the minimum number of users. The task coverage is defined as whether the total number of calls is equal to or more than a threshold at these 13 towers in a fixed time period. Their algorithm predicts the call probability of users to estimate the current coverage, and then allows to assign more participant before the task period ends to enhance the chance of full coverage. Most recently, Xiong *et al.* [11] further consider a task assignment problem under budget constraint, where the optimization goal is to maximize the number of calls (sensing data) for certain location sets under an overall budget constraint (both base and bonus incentives are given selected participants). Pournajaf *et al.* [15] also study task assignment in MCS aiming to assign moving participants with uncertain trajectories to static sensing tasks. The optimization goal is to minimize the coverage cost while maximize or maintain certain-level coverage (in term of the number of selected participants per target). The



(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Fig. 12. Performance of different caching size when $n = 300$, $\lambda = 200$, $\gamma = 0.5$ and $D = 300$ to 700.

coverage cost is based on the distance between the participant and the task location. But it only considers static spatial tasks (i.e. location-based tasks) which ignore the temporal requirements of sensing tasks. Overall, in all of these existing works, the sensing tasks are static with fixed time period and no new sensing task can come after the MCS starts. In [16], Li *et al.* study a dynamic participant selection problem, where heterogeneous sensing tasks can arrive at any time and may have various temporal/spacial requirements and with various sensing periods. Both offline and online algorithms are proposed to handle the dynamic problem.

Caching mechanism has been widely used in information technology, such as Web applications [17], P2P networks [18] and mobile computing [19]. Systems implemented with caching mechanism usually improve various type of performances by leverage the usage of cached data. Meanwhile, there are various caching strategies proposed and utilized for different systems with different characteristics. In this paper, we introduce caching into MCS and carefully design the new participant selection algorithms and corresponding caching strategies for such systems.

VI. Conclusion

In this paper, we introduce a new MCS system with caching capability, and study a dynamic participant selection problem for heterogeneous sensing tasks in such a system. The caching component enables new online participant selection algorithm which can predict the future tasks and dynamic assign participants based on estimated coverage improvement. In addition, when the caching space is full, a coverage based caching strategy can be used to make smart decision on which cached data to drop. Overall, the newly introduced mobile crowd sensing with caching can significantly use less selected participants to achieve similar level of probabilistic coverage than the previous best solution without caching. This is confirmed by extensive simulations conducted with real-life D4D dataset. We leave further improvements on call prediction as one of our future works.

Acknowledgement: The work is partially supported by the US National Science Foundation under Grant No. CNS-1319915 and CNS-1343355, and the National Natural Science Foundation of China under Grant No. 61370192, 61428203, 61432015 and 61572347. The authors would like to thank Orange and the D4D challenge organizers for providing them the D4D datasets and allowing them to continue working on the datasets after the challenge. F. Li and Y. Wang are co-corresponding authors.

References

[1] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Yen, R. Huang, et al., "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comp. Surveys*, 48(1), e7, 2015.
[2] L. Bengtsson, X. Lu, A. Thorson, R. Garfield, and J. von Schreeb, "Improved response to disasters and outbreaks by tracking population movements with mobile phone network data: A post-earthquake geospatial study in Haiti," *PLoS Med*, 8(8):e1001083, 2011.
[3] P. Zhou, et al., "How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing," in *ACM MobiSys*, 2012.
[4] C. Bo, T. Jung, et al., "SmartLoc: Sensing landmarks silently for smartphone based metropolitan localization," *EURASIP Journal on Wireless Communications and Networking*, 2016(1)1-17.
[5] C. Bo, X. Jian, T. Jung, J. Han, X.-Y. Li, X. Mao, and Y. Wang, "Detecting driver's smartphone usage via non-intrusively sensing driving dynamics," *IEEE Internet of Things Journal*, to appear.
[6] S. Nawaz, C. Efstratiou, et al., "Parksense: A smartphone based sensing system for on-street parking," in *Proc. of ACM Mobicom*, 2013.
[7] R. K. Rana, C. T. Chou, et al., "Ear-phone: An end-to-end participatory urban noise mapping system," in *Proc. of ACM/IEEE IPSN*, 2010.
[8] M. Mun, S. Reddy, et al., "PEIR, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proc. of ACM MobiSys*, 2009.
[9] N. Lathia, V. Pejovic, K. Rachuri, C. Mascolo, M. Musolesi, and P. Rentfrow, "Smartphones for large-scale behavior change interventions," *IEEE Pervasive Computing*, 12(3):66–73, 2013.
[10] A. Noulas, S. Scellato, R. Lambiotte, M. Pontil, and C. Mascolo, "A tale of many cities: universal patterns in human urban mobility," *PLOS ONE*, 7(5): e37027, 2012.
[11] H. Xiong, D. Zhang, L. Wang, J. Gibson, and J. Zhu, "EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants," *ACM Trans. on Intelligent Sys. and Tech.*, 6(3): Article 39, 2015.
[12] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *Proc. of IEEE Percom*, 2015.
[13] D. Zhang, H. Xiong, L. Wang, and G. Chen, "Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proc. of ACM UbiComp*, 2014.
[14] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi, "EMC^3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint," *IEEE Trans. on Mobile Computing*, 14(7):1355–1368, 2015.
[15] L. Pournajaf, L. Xiong, and V. S. Sunderam, "Dynamic data driven crowd sensing task assignment," in *Proc. of ICCS*, 2014.
[16] H. Li, T. Li, Y. Wang, "Dynamic participant recruitment of mobile crowd sensing for heterogeneous sensing tasks," in *Proc. of IEEE MASS*, 2015.
[17] S. Podlipnig and L. Boszormenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys*, 35(4):374–398, 2003.
[18] A.S. Vijendran and S. Thavamani, "Survey of caching and replica placement algorithm for content distribution in peer to peer overlay networks," in *Proc. of ACM CCSEIT*, 2012.
[19] D. Elsharief, H. Ibrahim, et al., "A survey of methods for maintaining mobile cache consistency," in *Proc. of ACM MoMM*, 2009.
[20] V. D. Blondel, et al., "Data for development: The D4D challenge on mobile phone data," in *arXiv.1210.0137v2*, 2013.