PARTICIPANT SELECTION AND TASK ASSIGNMENT IN MOBILE CROWD SENSING

by

Hanshang Li

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2018

Approved by:

Dr. Yu Wang

Dr. Weichao Wang

Dr. Dazhao Cheng

Dr. Tao Han

ABSTRACT

HANSHANG LI. Participant Selection and Task Assignment in Mobile Crowd Sensing. (Under the direction of DR. YU WANG)

With the rapid increasing of smart mobile devices and the advances of sensing technologies, mobile crowd sensing (MCS) becomes a new popular sensing paradigm, which enables a variety of large-scale sensing applications. One of the key challenges of large-scale mobile crowd sensing systems is how to effectively select appropriate participants from a huge user pool to perform various sensing tasks while satisfying certain constraints. This becomes more complex when the sensing tasks are dynamic (coming in real time) and heterogeneous (having different temporal and spatial requirements).

In this work, we consider multiple participant recruitment problems in MCS. We firstly consider a dynamic participant recruitment problem with heterogeneous sensing tasks, which aims to minimize the sensing cost while maintaining certain level of probabilistic coverage. Both offline and online algorithms are proposed to solve the challenging problem. Then 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. A newly designed participant selection with caching is presented. We further investigate the feasibility of collecting data packets from mobile devices through device-to-device communications by carefully selecting the subset of relaying devices. We formulate the problem as an optimization problem and propose a simple solution to solve it in a large-scale mobile environment. While online learn-

ing techniques can be used to learn the participants capability, the diverse expertise of each individual makes a single capability metric is not sufficient. To address the multi-expertise of participants, we introduce a new self-learning architecture, which leverages the historical performing records of participants to learn the different capabilities (both sensing probability and time delay) of participants. Formulating the participant selection problem as a combinational multi-armed bandit problem, we present an online participant selection algorithm with both performance guarantee and bounded regret. Finally, we introduce the cumulative participant selection problem with switch costs and propose a corresponding online learning method. For each of the work above, extensive simulations with real-world mobile datasets are conducted for the evaluations of the proposed methods. Our simulation results confirm the effeteness of them.

ACKNOWLEDGMENTS

First and foremost, I would like to express the deepest appreciation to my advisor, Dr. Yu Wang for his selfless support and valuable guidance of my study and related research for years. He continually and convincingly conveyed a spirit of adventure in regard to research and scholarship. This dissertation can not be done without his guidance and persistent help.

Second, I am grateful to the service of all of those in dissertation committee. Each of the members of my Dissertation Committee has provided me extensive personal and professional guidance and taught me a great deal about both scientific research and life in general. In addition, I want to thank my team members in Wireless Networking and Sensing (WiNS) Lab for their excellent cooperation and unending inspiration.

I would also like to thank my parents and my wife Huifang for their wise counsel and sympathetic ear. Finally, I would like to thank all my friends for their company and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The widespread availability of smart phones equipped with built-in sensors has enabled a new sensing paradigm, *mobile crowd sensing* (MCS), where tremendous data can be obtained by the large group of selected mobile participants over a wide geographical region [33]. Compared with traditional static sensor networks, MCS leverages existing sensing and communication infrastructures without additional costs; provides unprecedented spatio-temporal coverage, especially for observing unpredictable events; and integrates human intelligence into the sensing and data processing. These advantages has enabled a broad range of MCS applications, such as public safety [10, 6] , traffic planning [12, 14], localization [40, 41], environment monitoring [16, 17], and urban dynamic mining[8, 9].

While large-scale mobile crowd sensing system takes the advantage of huge number participants to enable massive mobile data sensing within urban environments, it also brings many new challenges in the system design. One of the key challenges is participant selection problem, how to effectively select appropriate participants from a huge user pool to perform various sensing tasks while satisfying certain constraints. On one hand, more selected participants in MCS can lead to better coverage of sensing tasks over both temporal and spacial domains. On the other hand, the overall sensing cost needs to be minimized, since performing sensing task is not free (costs energy of the smart phone). This cost could be as simply as the number of selected

participants when the cost per participant is fixed. Therefore, careful design of participant selection scheme becomes crucial, especially in large-scale MCS system, for the overall performance of crowd sensing and its associated cost.

Recently, there are several studies [18, 19, 20, 21, 22, 23] 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. However, some of the methods (such as [22, 23]) only consider the spacial tasks (requiring the coverage of a set of static interested points in spacial domain) and ignore the possibility of temporal requirements of sensing tasks. Some of the other methods (such as [18, 19, 20, 21]) do consider both temporal and spacial coverage requirements but they assume that the sensing tasks are static (generated before the starting of MCS and no further tasks can come after MCS starts). In reality MCS sensing tasks are heterogeneous, they can have different temporal and spacial requirements and various sensing periods. More importantly, the sensing tasks could arrive at any time. Therefore, new dynamic participant selection methods are needed for such MSC system with heterogenous sensing tasks.

In our first work, we formulate a new dynamic participant recruitment problem with heterogeneous sensing tasks in a large-scale piggyback MCS system. In a piggyback MCS system[18, 19], the collected sensing data returns to the system by leveraging smartphone usage opportunities to save energy consumption. Therefore, we only focus on the participant recruitment part. We show that finding the minimum participants to achieve certain level of coverage of all tasks is a very challenging problem (actually

a NP-hard problem). We then carefully design three greedy algorithms (one offline and two online) to tackle the dynamic participant recruitment problem. Our results show the proposed methods can achieve stable task coverages while use less number of participants against other simple solutions. We believe that this study is the first on dynamic participant recruitment with heterogeneous sensing tasks in MCS systems.

In our second work, we introduce a data storage component into the MCS system (as shown in Figure 1) such that the sensing data can be cached to fulfill future incoming tasks. Caching mechanism has been widely used in many networking systems [35, 36, 37]. 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 third work, we focus on the data collection phase of mobile data sensing by carefully selecting a few mobile participants as relay nodes to help with data propagation via D2D relays. By doing so, we limit the search space and make our algorithm more efficient. In addition, since we use multiple space-time paths for data collection from the source (in [50] only one space-time path is selected for one source), our method can achieve better delivery ratio too. We also consider the joint problem where the selected participants perform both sensing and data collection.

In our last two works, we formulate and investigate a dynamic *participant selection problem* (PSP) and *cumulative participant selection problems* (with or without switch costs) respectively in this work. In our MCS model, the selection mechanism

selects the participants for heterogeneous sensing tasks on both spatial and temporal domain. The selected participant has unknown probability to perform the assigned sensing tasks. After performing sensing tasks, the participant may utilize different ways to upload collected sensing data, such as WiFi, Device-to-Device (D2D) or cellular networks. Different uploading methods may lead different delay time, which is still unknown by the selection mechanism. Having the historical records of participant performing tasks, we propose online learning algorithms to solve the participant problem by leveraging combinational multi-armed bandit (MAB) concepts.

With the completed works introduced above, we also discuss our future work in this dissertation. We will mainly focus on online learning mechanisms which could enhance the participant selection and task assignment in mobile crowd sensing systems.

In summary, in this dissertation we study on the participant selection and task assignment in mobile crowd sensing. The rest of this dissertation is organized as follows. We first introduce the backgrounds and related works of this problem in Chapter 2. Then we present the works mentioned above in three sections respectively. Finally, Chapter 8 concludes this paper and presents the future works and the timeline.

CHAPTER 2: BACKGROUNDS RELEVANT TO PROPOSED WORK

2.1 Participant Selection and Task Assignment

With the wide adaptation of mobile crowd sensing applications[7, 33], task coverage and participant selection in MCS system has drawn many attentions from researchers in recent three years. First, there are several system and experimental studies on either experimental study on MCS coverage [26] or general framework of participant recruitment[27, 28]. For example, Chon *et al.* [26] has preformed a systematic study of the coverage and scaling properties of place-centric urban crowd sensing and shows promising results that MCS can provide relatively high coverage levels especially given area with large size. Then, there are also many theoretical studies on various task assignment and participant selection problems, playing tradeoffs among sensing cost, task coverage, energy efficiency[21, 29, 57], user privacy[23], and incentive[30, 31, 32, 31]. In most cases, task assignment is equivalent to participant selection. In this section, we only focus on reviewing those who are directly related to our work.

Pournajaf *et al.* [22] 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 coverage cost is based on the distance between the participant and the task location. An adaptive framework is

also proposed to refine the trajectories and perform local task refinement at selected participants. However, no detailed task assignment algorithms are proposed. Then, the same authors also apply similar idea to perform spatial task assignment with cloaked locations to protect the users' privacy in [23], and propose a two-stage task assignment method. However, both these works only consider static spatial tasks (i.e. location-based tasks) which ignore the temporal requirements of sensing tasks and do no allow dynamic tasks.

2.2 Data Collection

Recently, with the advances in mobile opportunistic networks or delay tolerant networks [62, 59, 60, 61] and D2D offloading [45, 46, 47], D2D data collection for mobile sensing becomes a new trend. Wang *et al.* [49] first consider leveraging the delay-tolerant mechanisms by offloading the data to Bluetooth/WiFi gateways or data-plan users. Their objective is to reduce the energy consumption and data cost of data-plan users. Karaliopoulos *et al.* [50] consider a joint user recruitment problem for both sensing and data collection, which is very similar to the one we study since the data collection is also done via D2D communications. However, the selection of users is formulated as a minimum cost set cover problem and single greedy heuristics are proposed to solve it. Since the solution space over all space-time paths is huge, their method may not be suitable for large-scale data collection. In this work, instead we carefully selects a few mobile participants as relay nodes to help with data propagation via D2D relays. By doing so, we limit the search space and make our algorithm more efficient. In addition, by leveraging multiple space-time paths for

data collection, our method can achieve better delivery ratio too.

2.3 Learning Based Selection

There are a few studies begin to focus on learning the participant capability for different sensing tasks so that better participant selection can be achieved [71, 72, 73]. For example, Han et al. [71] first adopt an online learning approach to acquire the statistical information about the sensing values from participants throughout the selection process. In their model, the quality of sensing data acquired by the participants are uncertain as random variables and MCS selection aims to maximize its expected total sensing revenue under a limited budget. They proved the proposed method has a asymptotically optimal regret bound. However, they only considers the homogeneous sensing tasks, which limits their applications in many cases. Liu and Liu [72] focus on the online labeling problem in which the true label is unknown. Since the labeling outcome cannot be directly verified, it can only be estimated against the crowd probabilistically. They proposed an online algorithm using majority voting rule to differentiate high and low quality labelers over time and proved that their method has a bounded regret under mild assumptions on the collective quality of the crowd. However, it only considers homogeneous sensing tasks with one-dimension metric. Zhang et al. [73] also consider expertise-aware task allocation and truth analysis in MCS where user expertise is estimated via a general online learning framework. In their methods, the participant expertise is learned based on semantic analysis. Though it considers heterogeneous tasks in a semantic manner, it may not be easily adopt to the sensing tasks with diverse temporal and spatial requirements. In addi-

tion, none of these learning methods consider either the process of data collection or the possible switch cost. In stead, in this work, we aim to provide new self-learning method for participant selection mechanism to handle heterogeneous sensing tasks, diverse data collection methods, and possible additional participant switch cost.

CHAPTER 3: DYNAMIC PARTICIPANT RECRUITMENT FOR HETEROGENEOUS SENSING TASKS IN MOBILE CROWD SENSING

3.1 Introduction

In this work, we formulate a new dynamic participant recruitment problem with heterogeneous sensing tasks in a large-scale piggyback MCS system. In a piggyback MCS system[18, 19], the collected sensing data returns to the system by leveraging smartphone usage opportunities to save energy consumption. Therefore, we only focus on the participant recruitment part. We show that finding the minimum participants to achieve certain level of coverage of all tasks is a very challenging problem (actually a NP-hard problem). We then carefully design three greedy algorithms (one offline and two online) to tackle the dynamic participant recruitment problem. Note that since we cannot foreknow when and where a participant will place a phone call during the real crowding sensing period, our proposed methods are based on data driven solution which leverages knowledge obtained via historical call and location traces. We conduct extensive simulations over a real-life mobile dataset (D4D data set[24]) to evaluate the proposed algorithms in different MCS settings. Our results show the proposed methods can achieve stable task coverages while use less number of participants against other simple solutions. We believe that this study is the first on dynamic participant recruitment with heterogeneous sensing tasks in MCS systems.

3.2 System Model and Problem Statement

3.2.1 System Model

The mobile crowd sensing system includes the following components: a huge number of *mobile participants* who are willing to perform sensing tasks assigned to them, a set of *crowd sensing applications* who are continuously generating *crowd sensing tasks* and looking for sensing data from assigned participants, and the proposed *participant recruitment* component which dynamically decides particular participants for each sensing task. Figure 1 illustrates the overall framework. In this work, we assume that the task assignment can be sent to each selected participant via cellular service at any time, while the sensing data collected by selected participants will be piggybacked to the mobile crowd sensing system as in [18, 19]. Therefore, we will only focus on the participant selection process.

We assume there are n participant candidates (smartphone users who are registered for participating sensing tasks), denoted as $U = \{u_1, \cdots, u_n\}$, and o sensing locations, denoted by $L = \{l_1, \cdots, l_o\}$. Each user u_i has his own mobility pattern over temporal and spacial domain, which can be described as a predication probability $p(u_i, l_j, t)$, that is the probability of user u_i to place a phone call (or sensing data) at location l_j at time slot t within the whole sensing cycle T (e.g., one or two weeks). For example, Figure 2(a) show three call probability matrices $p(u_i, l_j, t)$ for three users.

A set of m heterogeneous *crowd sensing tasks* $S = \{s_1, \cdots, s_m\}$ generated from the crowd sensing applications. Each of the sensing task s_k can arrive at the system at any time from 1 to T, and we assume that $t_0(s_k)$ is the time slot s_k arrives, and tasks

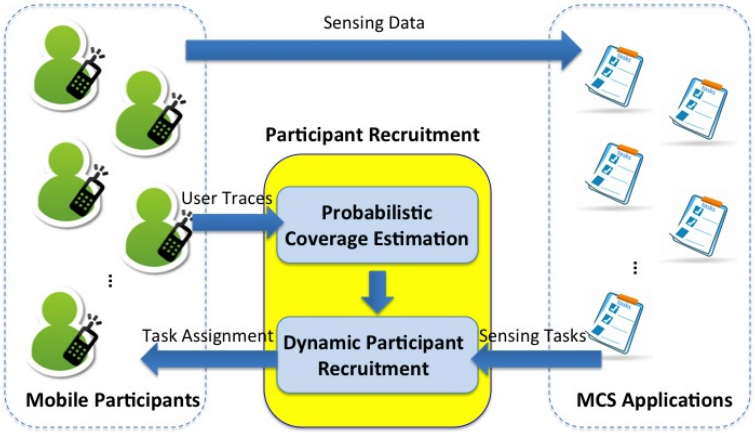


Figure 1: The framework of dynamic participant recruitment for MCS.

in S are ordered by $t_0(s_k)$. Each task specifies a set of targets for data collection, which includes a location l, a starting time t_s, and an ending time t_e of each target. For each target, the task aims to find a participant to sense the data at location l between time t_s and t_e. In other word, for piggyback crowd sensing if a selected user makes a call at l within the time of $[t_s, t_e]$, we consider that this target is covered or accomplished. To simplify, here we assume that each task only has one single target and the duration of the task (also called life time of the task) is bounded by a fixed value τ (say one day), i.e., $t_e - t_s \leq \tau$. Notice that our proposed methods can be easily extended to handle where sensing tasks with multiple targets. Let $t_s(s_k)$, $t_e(s_k)$, and $l(s_k)$ represent the time and location requirements of task s_k. Figure 2(b) shows three tasks with various lengths in different colors. Then we are ready to formally define the participant recruitment problem.

3.2.2 Participant Recruitment Problem

Given the pool of candidates U and the crowd sensing tasks S, the participant recruitment problem aims to minimize total sensing cost while still satisfying certain

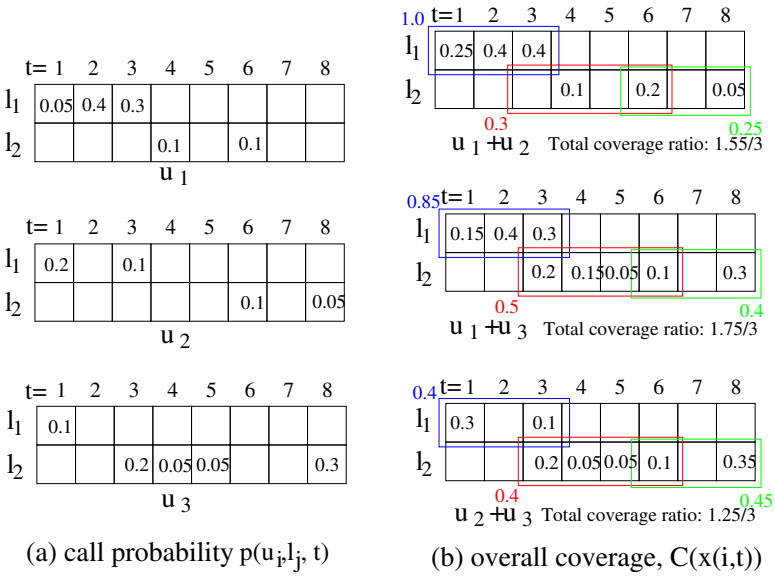


Figure 2: An example with 3 users (u_1, u_2, and u_3), 2 locations (l_1 and l_2), and 3 tasks (with various temporal and spacial coverage marked as blue, red and green rectangles). (a) call probability matrix of each user; (b) coverage ratio of any two users over these three tasks. The colored numbers at corners of tasks are the overall coverage ratios.

level of probabilistic coverage of the tasks. The output of the participant recruitment is a set of selected participants to perform the tasks shown by an indicator $x(i,t)$ where $x(i,t) = 1$ if user u_i is selected to participate starting from time t, otherwise $x(i,t) = 0$. Here we assume that when a mobile user is selected to perform sensing task at t, it could cover a fixed time period of τ (e.g., one day). Therefore, we restrict the participant selection of the same user within τ as follows:

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

Note that a single selected participant can perform the sensing task for multiple tasks and can also be selected multiple times at different time. Then we can define the cost

of sensing task as the summation of all selected participants:

$$\sum_{i\in[1,n]}\sum_{t\in[1,T]} x(i,t).$$

Here the total number of selected participants reflects the total sensing cost (a fix cost per selected participant, such as energy cost of being active for τ). Overall, we would like to minimize the sensing cost.

On the other hand, we also care about the coverage of the sensing tasks. If a selected user u_i makes a call at location $l(s_j)$ within the time of $[t_s(s_j), t_e(s_j)]$, we consider that this task s_j is covered and accomplished. Let $C(i,j,t)$ be the coverage ratio of task s_j by user u_i who is selected starting from t. Then the coverage of task s_j is defined as follows:

$$\min(\sum_{t\in[1,T]}\sum_{i\in[1,n]} C(i,j,t)x(i,t), 1) \textit{ for any } j \in [1,m].$$

Note here if multiple selected users cover the same task, the coverage ratio cannot exceed 1, i.e., fully covered. For example, in Figure 2(b), if u_1 and u_2 are selected, the coverage ratio of the blue task will be 1 even thought the summation of all coverage ratio is larger than 1. Since we cannot foreknow when and where a participant will place a phone call during the crowding sensing period T, we will estimate the $C(i,j,t)$ based on historical call and location traces. We can also define the overall coverage ratio of all task as follows:

$$C(x(i,t)) = \frac{\sum_{j=1}^{m}\min(\sum_{t\in[1,T]}\sum_{i\in[1,n]} C(i,j,t)x(i,t), 1)}{m}.$$

The overall coverage constraint is not a full coverage requirement, instead a prob-

abilistic coverage requirement (i.e., total task coverage is equal to or larger than a predefined coverage threshold γ).

Definition 1. Given the volunteering users U (with their historical call and location traces) and the crowd sensing tasks S, the *Participant Recruitment Problem* is to find participants (i.e., $x(i,t)$) with the objective to

$$\min_{x} \sum_{i\in[1,n]} \sum_{t\in[1,T]} x(i,t)$$

$$s.t.\ \ C(x(i,t)) \geq \gamma$$

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

$$x(i,t) = 0 \textit{ or } 1 \textit{ for any } t \in [1,T], i \in [1,n].$$

Figure 2 shows an example with three users and three tasks. In Figure 2(b), the coverage provided by any two of the three users is provided. It is obvious that choosing u_1 and u_3 leads to best coverage among these three choices. When the numbers of users and tasks are huge, solving this newly defined *participant recruitment problem* (PRP) is a computationally difficult task even when $C(i,j,t)$ is known. We can prove that this problem is NP-hard.

Theorem 1. The *Participant Recruitment Problem* (PRP) is NP-hard.

Proof. This can be obtained from the reduction of the minimum set cover (MSC) problem. Given an instance of MSC, we can construct an instance of PRP as follows. For the set of elements in MSC, we treat them as locations in PRP. Then for each of the subsets in MSC, we create a mobile user who can visit the locations whose

corresponding elements are within this subset. Only one sensing task is defined as visiting all locations. Let $T = \tau = 1$ and $\gamma = 1$, we now have a PRP constructed where its optimal solution provide a optimal solution of MSC. Such construction can be done in polynomial time. Since MSC is a well-mown NP-hard problem, PRP is also NP-hard. □

3.3 Participant Recruitment Algorithms

In this section, we introduce our proposed participant recruitment algorithms for PRP. 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. Such an assumption is reasonable for large-scale crowd sensing. We first show how we estimate the call probability $p(u_i, l_j, t)$ of a particular user and predict the task coverage ratio $C(i, j, t)$ for any task based on a data driven approach.

3.3.1 Estimation of Call Probability and Coverage Ratio

The call probability $p(u_i, l_j, t)$ of a particular user u_i to make a phone call at location l_j and time t is a critical and necessary knowledge for participant recruitment. 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 learning from the historical call and location traces. Here, we assume that for each user we have multiple rounds of call traces (e.g., K weeks) in the historical data, and each round of data denoted as D_i, $i = 1, \cdots, K$. Let $X_k(u_i, l_j, t)$ represent whether user u_i made one or more phone call at location l_j and time t in D_i (1 if it made, 0 otherwise).

Then we simply estimate the call probability as follow,

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

Note that we do not consider more complex models where location-transition process is modeled by either Bayesian interferon or Markov model [22] or the call sequence is followed as an inhomogeneous Poisson process [18, 19]. However, such models can be easily integrated into our framework.

In our proposed participant recruitment algorithms, in each round the coverage ratio $C(i, j, t)$ of task s_j by user u_i starting from t is estimated so that we can have a criteria to select individual user. Therefore, we now introduce how we estimate $C(i, j, t)$ from the call probability obtained from historical data. For offline algorithms, the coverage ratio is a summation of the call probability of each time unit within the sensing duration τ.

$$C(i, j, t) = \min(\sum_{t'=t}^{t_e(s_j)} p(u_i, l(s_j), t'), 1). \tag{1}$$

In addition, for proposed online algorithms, we also need to estimate the possible coverage $C(i, j, t)$ of the remaining active task s_j from a selected or potential user u_i.

$$C(i, j, t) = \begin{cases} 1 & \text{a call in } [t_s(s_j), t] \\ \min(\sum_{t'=t}^{t_e(s_j)} p(u_i, l(s_j), t'), 1) & \text{no call yet} \end{cases} \tag{2}$$

Note that if a selected user u_i already made a call between $t_s(s_j)$ and current time t as shown in Figure 3(a), task s_j has been covered by u_1 so $C(i, j, t) = 1$. Otherwise, without any call so far, u_i's contribution to task s_j is calculated for the remaining time of this task (as shared areas shown in Figure 3(b) and (c)).

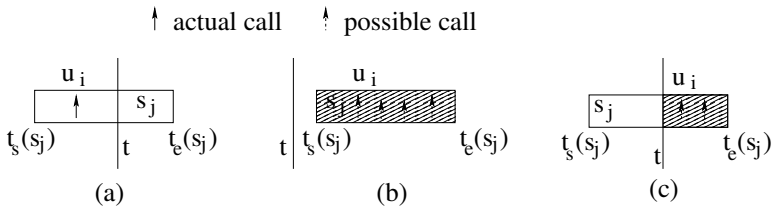


Figure 3: Three cases of estimation of $C(i, j, t)$ for user u_i and task s_j.

3.3.2 Offline Algorithms

Now we are ready to describe our basic offline greedy algorithm. Here we assume that the algorithm knows the whole set of task S for the whole sensing period T. Then in each round we add one participant into the selected pool by greedily selecting the one with largest increasing of total coverage ratio. For example, there are two candidate users u_1 and u_2 (with call probability shown in Figure 4(a) and (b)) and two tasks (with current coverage from previous selected users shown in Figure 4(c)). The offline algorithm will estimate the coverage ratio if select one of the users (as shown in Figure 4(d) and (c)) and pick the one with larger coverage (i.e., u_1 in this example). Algorithm 1 gives the detail of the agorithm. The time complexity of this algorithm is $O(n^2mT^2)$ since at most nT improvements are tested at each round, each improvement involves at most m tasks, and there are at most nT rounds.

Notice that we can also replace Line 4 of the greedy algorithm with other criteria, such as the most active user with maximal calls or even pure random choice. We test those offline methods in the simulation section too.

Algorithm 1 Offline Participant Recruitment Algorithm

Input: participant pool U, task set S, and call probability $p(u_i, l_j, t)$ for each user in U.

Output: $x(i,t)$.

1: $x(i,t) = 0$ for all i and t
2: **while** $C(x(i,t)) < \gamma$ **do**
3: **for all** $u_i \in U$ and $t \in [1,T]$ and $x(i,t) = 0$ **do**
4: Calculate the improvement of $C(x(i,t))$ by adding u_i at time t, i.e., $x(i,t) = 1$ (Here, the coverage ratio is calculated based on Equation (1))
5: **end for**
6: Select the user u_i at time t who leads to the largest coverage improvement, and set $x(i,t) = 1$
7: **end while**
8: **return** $x(i,t)$

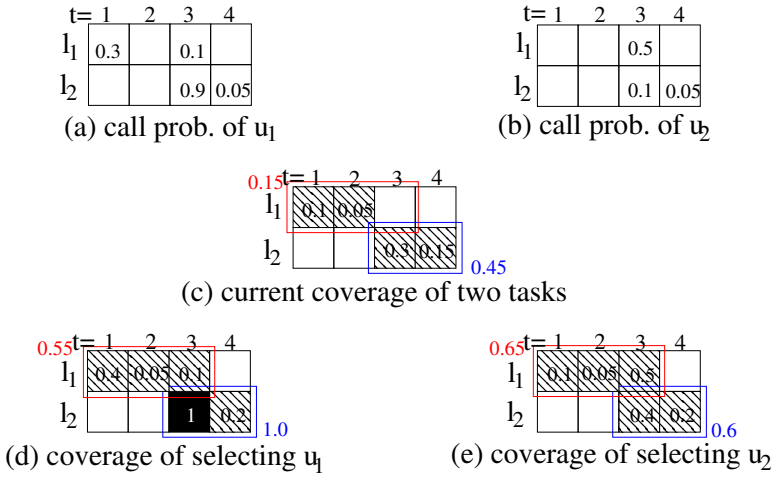


Figure 4: An example of user selection for two tasks (in red and blue). Shaded cell represents some level of coverage at this cell, while black cell represents full coverage of it. The colored numbers at corners of two tasks are the overall coverage ratios of those tasks.

3.3.3 Online Algorithms

In our offline algorithm, the coverage estimation is based on knowledge learned from historical data. However, whether a user make a call at the real sensing period is a random event and the prediction could be wrong. Thus, the actual coverage ratio could be much less than our estimation. One possible enhancement is to add more

Algorithm 2 Online Participant Recruitment Algorithm with Whole Task Set S

Input: participant pool U, task set S, and call probability $p(u_i, l_j, t)$ for each user in U.

Output: $x(i,t)$.

```
1: x(i,t) = 0 for all i and t
2: for t' = 1 to T do
3:    while C(x(i,t)) < γ do
4:       for all u_i ∈ U and t ∈ [t', T] and x(i,t) = 0 do
5:          Calculate the improvement of C(x(i,t)) by adding u_i at time t, i.e., x(i,t) =
            1 (Here, the coverage ratio is calculated based on Equation (2))
6:       end for
7:       Select the user u_i at time t who leads to the largest coverage improvement,
         and set x(i,t) = 1
8:    end while
9: end for
10: return x(i,t)
```

participants whenever the estimated sensing coverage is lower than the required ratio. Notice that when time goes by, the coverage ratio of a user to a task may change (as shown in Equation (2), depending on the current time and the remaining duration of the task or whether a call already being made). Therefore, at each time step, our online algorithm will check whether the task has been fulfilled by the current participants. If not, it will greedily selects the new participant who can maximize the coverage improvement (by considering the dynamic of coverage ratio). The details of this algorithm is described in Algorithm 2. The time complexity of this algorithm is $O(n^2mT^3)$. Clearly, this online algorithm needs more selected participants since it adds more participants when the tasking ending time is near and such a task is still not fulfilled.

So far, we always assume that the participant recruitment algorithm knows the whole set of sensing task S beforehand. However, in reality, the tasks are arriving at any moment and the algorithm may not know what kind of tasks will arrive in the

future. Therefore, we also provide a real online algorithm to handle such more practice scenario. See Algorithm 3 for detail. Here, at current time slot t, the algorithm will consider the current task set S_t which includes both previous started tasks and new tasks arrived at t. If the coverage ratio based on current selection does not reach the requirement, more participants will be selected in the same greedy fashion. Once again the coverage ratio is calculated based on on Equation (2) and only for tasks in S_t. Notice that even there is no new arriving task, the algorithm may still add more participants if the coverage of current tasks is not good enough. The time complexity of this algorithm for time t is $O(nm)$ since at most n users are selected at t and each user at most contributes to m tasks. It seems that this algorithm may not achieve the same level of coverage as Algorithm 2 since the decisions made here are without the knowledge of future incoming tasks. However, our simulation results show the opposite. This is due to that the pure online algorithm has to add enough participants to fulfill the coverage of current task set, which leads to more selected participants and also better coverage.

3.4 Simulations

In this section, we conduct extensive simulations over a real-life mobile data (D4D data set[24]) to exam the effectiveness of our proposed greedy algorithms under different participant recruitmen scenarios (e.g., online or offline). For the greedy criteria in both offline and online algorithms, we also implement a call activity based and a random one for the comparison with our proposed coverage-based solution. Thus, the following three greedy criteria are tested during the participant recruitment.

Algorithm 3 Online Participant Recruitment Algorithm with Current Task Set at Time t

Input: participant pool U, all previous selection $x(i, t')$ for $t' < t$, current task set S_t (including all tasks starting at or before t), and call probability $p(u_i, l_j, t)$ for each user in U.

Output: current selection $x(i, t)$.

1: Copy all previous selection to $x(i, t)$
2: **while** $C(x(i, t)) < \gamma$ based on S_t **do**
3: **for all** $u_i \in U$ and $x(i, t) = 0$ **do**
4: Calculate the improvement of $C(x(i, t))$ by adding u_i now at t, i.e., $x(i, t) = 1$ (Here, the coverage ratio is calculated based on Equation (2) and S_t)
5: **end for**
6: Select the user u_i who leads to the largest coverage improvement, and set $x(i, t) = 1$
7: **end while**
8: **return** $x(i, t)$

- Random: In each round, a random user is selected as the next participant of the MCS.
- Call: In each round, the user with highest call activity is selected as the next participant.
- Coverage: In each round, the user with largest coverage improvement is selected as the next participant.

In total, we implement seven participant recruitment algorithms: *Offline-Coverage* (Algorithm 1), *Offline-Call*, *Offline-Random*, *Online-Coverage* (Algorithm 3), *Online-Call*, *Online-Random*, and *Online-Coverage-TSK* (Algorithm 2) and compare their performance. Notice that *Online-Coverage-TSK* is the online algorithm with full knowledge of the tasks (including future tasks).

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

- Number of selected participants: the number of selected participants[1] generated by the algorithm for the whole task set over the sensing period.

- Number of fulfilled tasks: the number of sensing tasks which are successfully performed by selected participants from the algorithm during the sensing period.

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

3.4.1 D4D Dataset and Simulation Configuration

To simulate the large scale mobile crowd sensing (especially for mobile phone sensing), we use a real life wireless tracing data from the cellular operator Orange for the *Data for Development (D4D) challenge* [25]. The reason we pick the D4D dataset is that it is the only mobile networking tracing dataset available to us which has a large-scale and diverse set of mobile users. The released D4D datasets [24] are based on anonymized Call Detail Records (CDR) of phone calls and SMS exchanges between $50,000$ Orange mobile users in Ivory Coast between December 1, 2011 and April 28, 2012 (150 days and about 20 weeks). Most of the call records are generated between 6:00am to 11:00pm within each single day. 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. The time ranges of these 10 groups of records are sequential and add up equal to the whole duration of D4D data collection period. But unfortunately, in each group of records, the user IDs were renumbered

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

and anonymized, which makes impossible to merge them together. Thus, all of our MCS experiments are preformed 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.

For the D4D dataset, there are already huge number of users and encounters even within a two week period. For example, for the first two-week period, there are $46,254$ active mobile users, $1,097$ cellular towers, and $6,787,594$ encounters between users in total. Therefore, in our simulation, we only choose subsets of users as candidate participants and subsets of cellular towers as locations in MCS. For each sensing task s_i, we randomly pick its location $l(s_i)$, starting time $t_s(s_i)$ and ending time $t_e(s_i)$. For location $l(s_i)$, it is randomly chosen from 20 cellular towers with highest call records. Most of these towers are located in the region of Abidjan, the economic and former official capital of Ivory Coast and the largest city in the nation. Figure 5 shows the locations of these towers on the map of Abidjan. For starting time $t_s(s_i)$, it is randomly chosen from 1 to T. Then ending time $t_e(s_i)$ is randomly chosen from $t_s(s_i)$ to $t_s(s_i) + 24$. In other words, the duration of sensing period of a task is limited to one day. For candidate participants, we randomly choose them from the mobile users with the highest number of times of visiting the above towers. All parameters used in the simulations are shown in Table 1.

In all simulations, we randomly generate MCS tasks and apply different participant

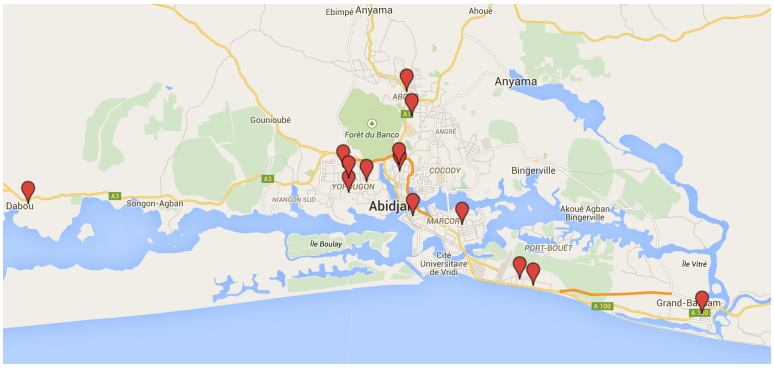
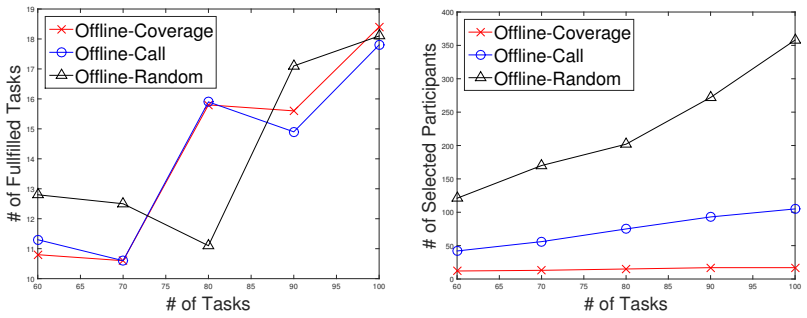


Figure 5: Locations of cellular towers near Abidjan used as sensing locations in our MCS simulations.

Table 1: Parameters used in simulation for PRP

Parameter	Value or Range
Unit of time	1 hour
Task life time $t_e(s_i) - t_s(s_i)$	1 to 24 hours
Number of locations (towers) o	20
Number of tasks m	$60, 70, 80, 90, 100$
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	$0.3, 0.4, 0.5, 0.6, 0.7$

recruitment algorithms to select participants for all tasks. The selected participant will 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 one call 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.



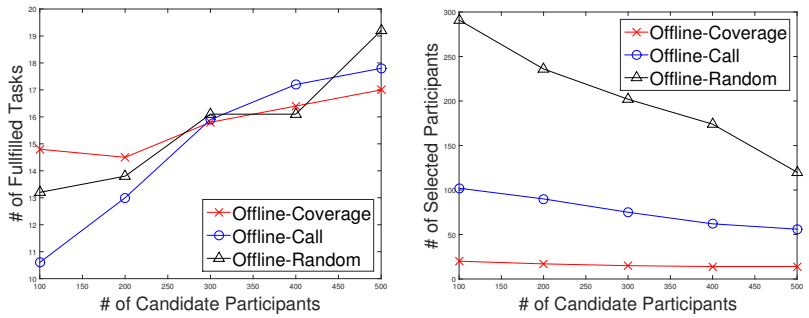
(a) (b)

Figure 6: Results of offline algorithms when $n = 300$ and $m = 60$ to 100.

3.4.2 Performance of Offline Algorithms

In the first set of simulations, we compare the performance among different greedy algorithms in the offline setting, in which the full task set S is known and the participant selection is performed offline without further updates. Here, we fix the number of candidate participant at 300, and vary the number of tasks from 60 to 100. Hereafter, the default $\gamma = 0.5$. Figure 6 shows the performance comparison of three offline algorithms.

Figure 6(a) shows the number of fulfilled tasks by each algorithm. Clearly, since the selection is based on historical data, the real coverage ratio cannot reach the expected level. However, for *Offline-Coverage* and *Offline-Call* have a clear pattern: the number of fulfilled tasks increases with the number of tasks. This reasonable since the coverage threshold is fixed. *Offline-Random* does not have this pattern. Figure 6(b) shows the number of selected participants. Obviously, the number of selected participants increases with the number of the tasks, i.e., more tasks need more participants. Compared with the three methods, *Offline-Random* uses the largest number of participants while *Offline-Coverage* has the minimum number of partic-

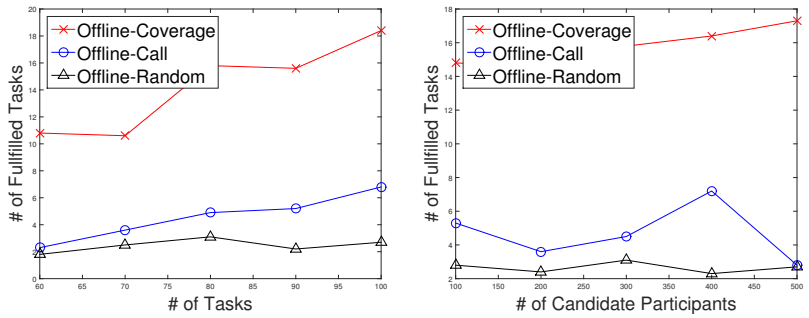


(a) (b)

Figure 7: Results of offline algorithms when $m = 80$ and $n = 100$ to 500.

ipants. The differences among these three methods are significant (*Offline-Random* or *Offline-Call* use 3 or 8 times more participants than *Offline-Coverage* does). This confirms the nice performance of our proposed offline algorithm. Interestedly, the number of participants of *Offline-Coverage* is always less than 20 no matter how many tasks we have. This shows the stability of the proposed method.

In the next set of simulations, we fix the number of task at 80 and test with various number of candidate participant. Figure 7 shows the result. It is clear that the number of fulfilled tasks increases with the number of candidate participants as shown in Figure 7(a), since more candidate participants offers more possible optimized selection. This observation is consist among all methods. The coverage achieved by the three methods are similar. However, in term of the number of selected participants, as shown in Figure 7(b), again *Offline-Coverage* uses much less participants than the other two methods to achieve the same level of coverage. In addition, with more candidate participants, the number of selected participants by all algorithms decreases. This is due to that more candidate participants lead to more space to make smart selections.

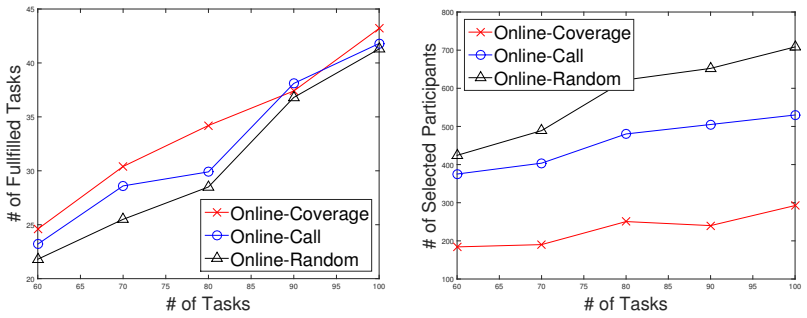


(a) $n = 300$ (b) $m = 80$

Figure 8: Results of offline algorithms with $n = 300$ or $m = 80$, where *Offline-Random* and *Offline-Call* are forced to select the same number of participants with *Offline-Coverage*.

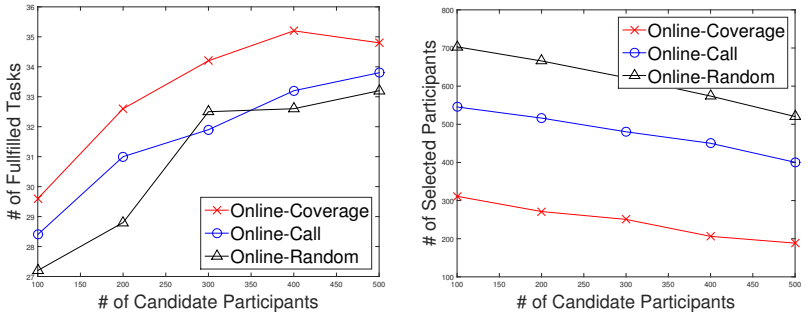
Via these two sets of simulations, we found that the task coverages (the numbers of fulfilled tasks) of the three offline methods are similar, but *Offline-Random* and *Offline-Call* select more participants to make up their low efficiency. We also implement another set of simulations to compare with their task coverage with the same number of selected participants. We use the number of selected participants of *Offline-Coverage* as the baseline, and force the other two methods select the same amount of participants. The results are reported in Figure 8. Clearly, now *Offline-Coverage* outperforms the other two methods in term of the number of fulfilled tasks. More precisely, the average numbers of fulfilled tasks of *Offline-Coverage*, *Offline-Call*, and *Offline-Random* are 14.24, 4.56 and 2.46 respectively for simulations with $n = 300$ and various values of m (Figure 8(a)); and 15.78, 4.68 and 2.66 respectively for simulations with $m = 80$ and various values of n (Figure 8(b)).

Overall, the proposed offline algorithm *Offline-Coverage* can achieve better performance than the other two methods. However, since our offline algorithm utilizes estimated call probability to predict the future calls, the achieved coverage ratio is



(a) (b)

Figure 9: Results of online algorithms when $n = 300$ and $m = 60$ to 100.



(a) (b)

Figure 10: Results of online algorithms when $m = 80$ and $n = 100$ to 500.

only one third of objective coverage threshold.

3.4.3 Performance of Online Algorithms

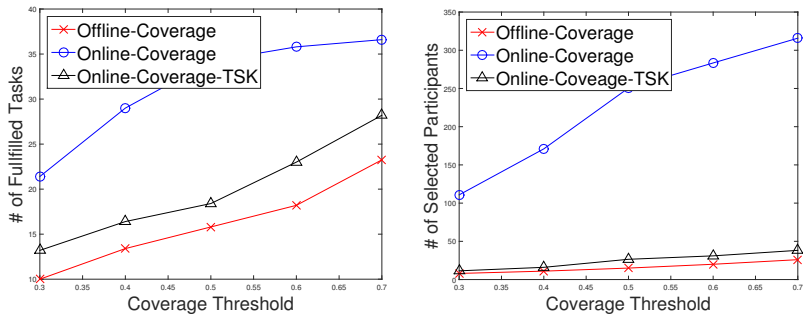
To further improve the achieved coverage ratio, we test our online algorithms under the pure online fashion, i.e., the algorithm only knows current sensing task set and have no knowledge of future tasks, but based on the current fulfilled status it can select more participants to achieve the desired coverage ratio. Here, we compare the three type greedy methods. The simulation settings are the same with those in offline tests.

Figure 9 and Figure 10 show the results over two set of simulations where either the number of participants or the number of task is changing. The overall conclusions are

similar to those in offline test. First, the number of fulfilled tasks of *Online-Coverage* almost always larger than those of the other two methods. More importantly, it uses much less number of selected participants to achieve such level of coverage. Numbers of selected participants in *Online-Random* and *Online-Call* are almost three times and two times of that in *Online-Coverage*. In addition, compared with offline-methods, online methods output more selected participants in the same task pool. This is due to additional participants are selected at any time to fulfill the unfinished tasks. This improves the coverage ratio but increases the number of selected participants.

3.4.4 Offline vs Online Algotihms

Last, we want to further study the difference among offline and online algorithms. Here, we also test the online algorithm with full knowledge of future tasks (i.e. *Online-Coverage-TSK* - Algorithm 2 in Section 3.3). Figure 11 shows the results of three algorithms (*Offline-Coverage*, *Online-Coverage*, and *Online-Coverage-TSK*) over the same sets of tasks with different coverage threshold γ (from 0.3 to 0.7). Here, $m = 80$ and $n = 300$. Overall, higher coverage threshold leads to higher number of fulfilled tasks and more selected participants. From Figure 11(a), the number of fulfilled tasks by *Online-Coverage-TSK* is about 50% more than the one by *Offline-Coverage*, while the one by *Online-Coverage* is about two times of that by *Online-Coverage-TSK*. Obviously, *Online-Coverage-TSK* finished the most number of tasks, but it also chooses the largest number of participants as shown in Figure 11(b). *Online-Coverage* selects many times of participants than that in the other two algorithms, while *Online-Coverage-TSK* chooses around 30% more participants than *Offline-Coverage* does.



(a) (b)

Figure 11: Results of offline/online algorithms when $m = 80$ and $n = 300$ with various values of γ.

Summary: From all of the simulation results above, we could draw the following general conclusions. Firstly, the coverage improvement is a better greedy criteria than those based on call activity or purely random selection. Secondly, online algorithms can achieve better coverage than offline algorithms, since they can actively select additional participants. Finally, there is a trade off between the number of selected participants and the coverage level of tasks. It should choose the right algorithm and selection strategy according to particular requirements from the MCS applications.

3.5 Summary

In this work, we focus on a new dynamic recruitment problem for heterogeneous mobile crowd sensing tasks, with a goal to minimizing the sensing cost while satisfying certain level of coverage. Unlike other existing works, the sensing tasks in our proposed scenario can have different starting time and life time. Based on the prediction of call probability (the probability of a user making calls at particular time and locations), we propose several offline and online greedy algorithms to dynamically select a subset of participant to perform the tasks. Via extensive simulations conducted

with real-life D4D dataset, we confirm the efficiency of our proposed algorithms. We leave further improvements on call prediction as one of our future works.

CHAPTER 4: ENHANCING PARTICIPANT SELECTION VIA CACHING IN MOBILE CROWD SENSING

4.1 Introduction

Caching mechanism has been widely used in information technology, such as Web applications [35], P2P networks [36] and mobile computing [37]. 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 work, we introduce caching into MCS and carefully design the new participant selection algorithms and corresponding caching strategies for such systems.

One of the uniquenesses of this study is that we introduce a data storage component into the MCS system (as shown in Figure 12) such that the sensing data can be cached to fulfill future incoming tasks. Caching mechanism has been widely used in many networking systems [35, 36, 37]. 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[24]) 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.

4.2 System Model and Participant Selection

4.2.1 System Model and Assumptions

As shown in Figure 12, 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

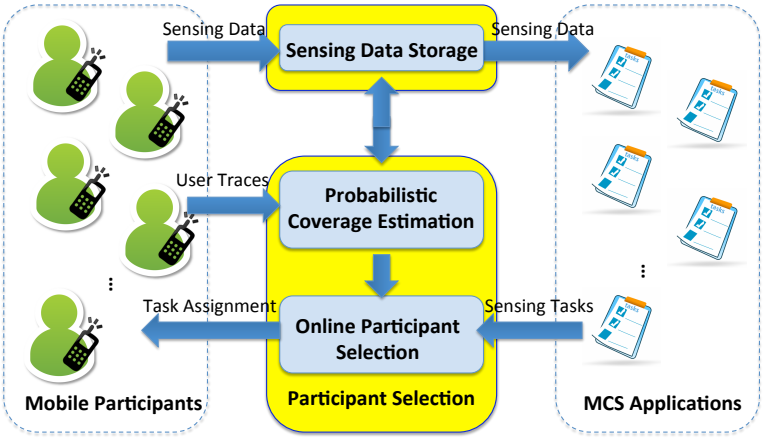


Figure 12: The architecture of MCS system with caching.

tasks.

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

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 work becomes crucial.

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 work, 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 [18, 19] during a phone call. Therefore,

we are interested in the call patterns than the visiting patterns[2]. 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 [22] or Poisson process [18, 19].

Based on this predicted information, the participant select mechanism can select a subset of participants to perform the sensing task. Here we use an indicator $x(i, t)$ 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

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

period of $[t, t+\tau]$. Therefore, we have to restrict the selection of the same participant within τ as follows:

$$\sum_{t'=t}^{t+\tau} x(p_i, t') \leq 1 \textit{ for any } t \in [1, T] \textit{ and } i \in [1, n]. \tag{3}$$

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, $\sum_{t\in[1,T]} x(p_i, t)$.

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.

4.2.2 Participant Selection Problem

Similar to [34], 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 $x(p_i, t)$. The overall optimization problem can be defined as:

$$\min_{x} \sum_{i\in[1,n]} \sum_{t\in[1,T]} x(p_i, t)$$

$$s.t.\ \ C(x(p_i, t)) \geq \gamma \ \textit{and}$$

$$\textit{Equation (1) on } x(p_i, t).$$

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. $C(x(p_i, t))$ 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 defined as follows:

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

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:

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

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 [34] for a simpler version of this problem). Therefore, in this work, 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.

4.3 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.

4.3.1 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,

$$\Delta(B, p_i, s_j, t) = C(B + p_i, s_j, t) - C(B, s_j, t). \tag{4}$$

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

$$\Delta(B, p_i, t) = \sum_{j=1}^{m} \Delta(B, p_i, s_j, t). \tag{5}$$

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

$$C(B, s_j, t) = 1 - \sum_{x=0}^{freq(s_j)-1} C^x(B, s_j, t). \tag{6}$$

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 (6) can be simplified to

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

4.3.2 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 which is associated 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[3]. 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:

$$p(s_j, t) = p(s_j, t_{begin}(s_j), l(s_j)n(t). \tag{7}$$

[3]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.

To assign participants to the possible future tasks or estimate the coverage ratio of them by particular participant set, we basically modify Equation (5) to the following.

$$\Delta(B, p_i, t) = \sum_{s_j \in S_t} \Delta(B, p_i, s_j, t) + \sum_{s_j \notin S_t} p(s_j, t)\Delta(B, p_i, s_j, t). \tag{8}$$

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.

4.3.3 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 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 4. In each round, the algorithm

Algorithm 4 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 $x(p_i, t)$

1: update the current task set $B_{t'-1}$ and previous selection $x(p_i, t)$ if there are new sensing data uploaded at data storage and partially fulfilling certain tasks from last time.
2: copy all previous selection $x(p_i, t)$ from $B_{t'-1}$.
3: **while** $C(x(p_i, t)) < \gamma$ 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 (8))
6: **end for**
7: Select the user p_i who leads to the largest coverage improvement, and set $x(i, t) = 1$
8: **end while**
9: **return** $x(p_i, t)$

basically repeatedly adding new participant which leads to largest coverage gain at current time until the estimated overall coverage reaches the requirement threshold.

4.3.4 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 4. 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 assign tasks that already arrived to the candidates. That means that the caching sensing data are 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 (5). 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 (8), 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 begin 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.

4.4 Simulations

In this section, we conduct extensive simulations over a real-life mobile traces (D4D data set[24]) to evaluate the effectiveness of our proposed participant algorithms under different scenarios.

4.4.1 D4D Dataset and Sensing Task Generation

To simulate the large scale mobile crowd sensing, we still utilize D4D datasets [24] to evaluate our proposed solution. 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. Figure 13 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,

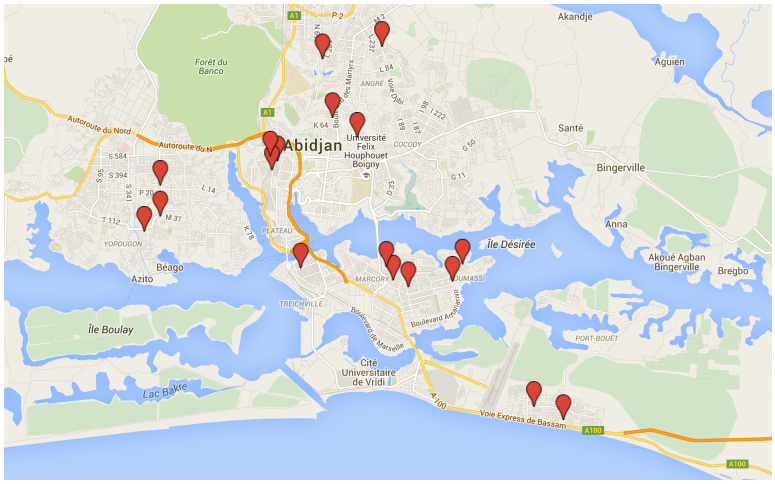


Figure 13: Locations of cellular towers in Abidjan used as sensing locations.

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 2.

In all simulations, we randomly generate MCS tasks based on the method discussed above, and apply different participant selection algorithms to select participants for

Table 2: Simulation parameters

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	$0.4, 0.45, 0.5, 0.55, 0.6$

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.

4.4.2 Tested Algorithms and Scenarios

Beside the proposed method, we also implement two simple algorithms and the one in [34] 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 [34].

- 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[4] generated by the algorithm for the whole task set over the sensing period.

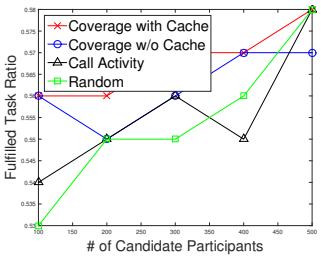
- Fulfilled task ratios: the ratio between the number of sensing tasks which are successfully performed by selected participants from the algorithm during the sensing period and the total tasks.

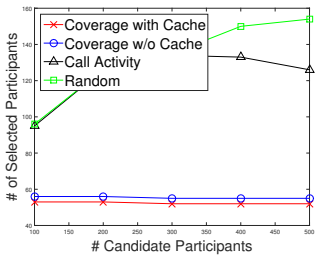
- Coverage ratio with caching: the portion of fulfilled tasks which are fulfilled by cached sensing data.

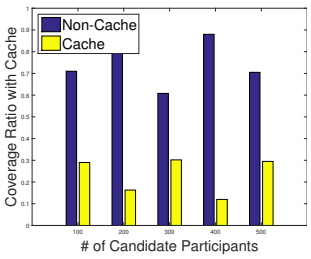
All results reported here are the average from multiple runs over different periods from the D4D data set.

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.

[4]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 $xp_(i, t_2) = 1$) and that is counted as multiple participants.

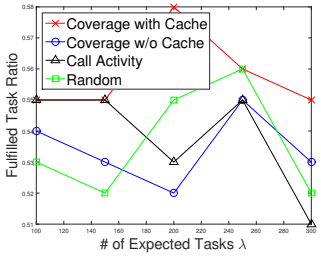


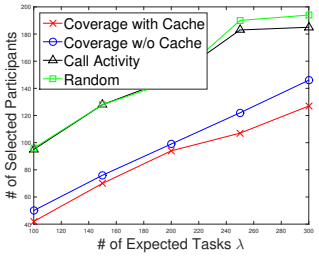


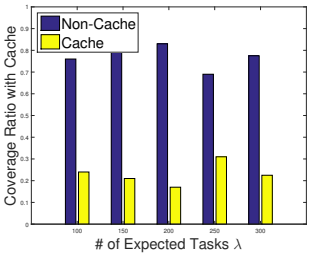


(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Figure 14: Results under Scenario A, when $\lambda = 200$, $\gamma = 0.6$ and $n = 100$ to 500.







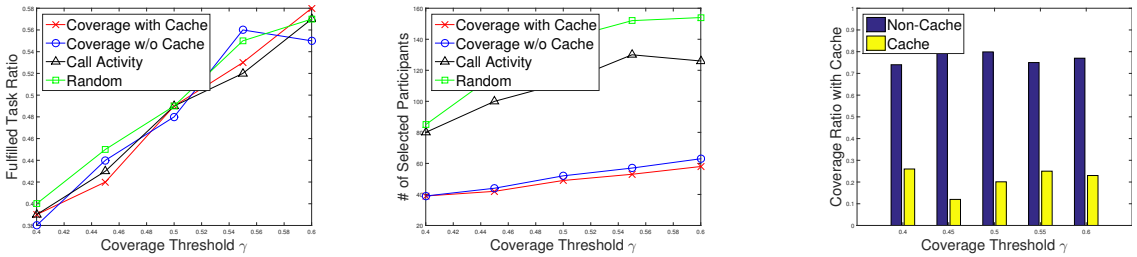
(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Figure 15: Results under Scenario A, when $n = 300$, $\gamma = 0.6$ and $\lambda = 100$ to 300.

4.4.3 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. Figure 14 shows the performance comparison of four different algorithms. Figure 14(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 Figure 14(b), the coverage based solutions have much fewer selected participants than the other two

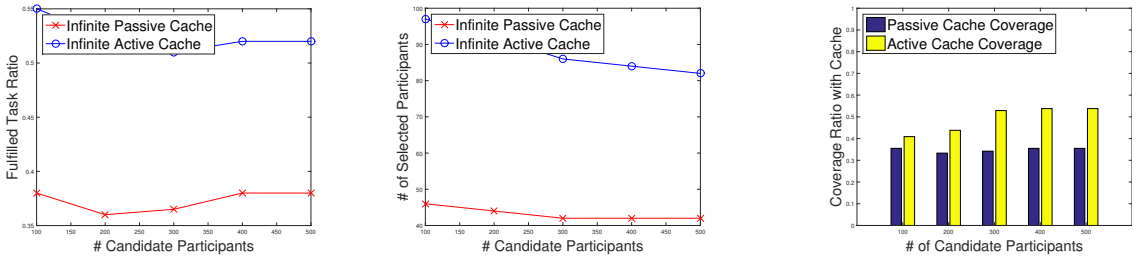


(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Figure 16: Results under Scenario A, when $n = 300$, $\lambda = 200$ and $\gamma = 0.4$ to 0.6.

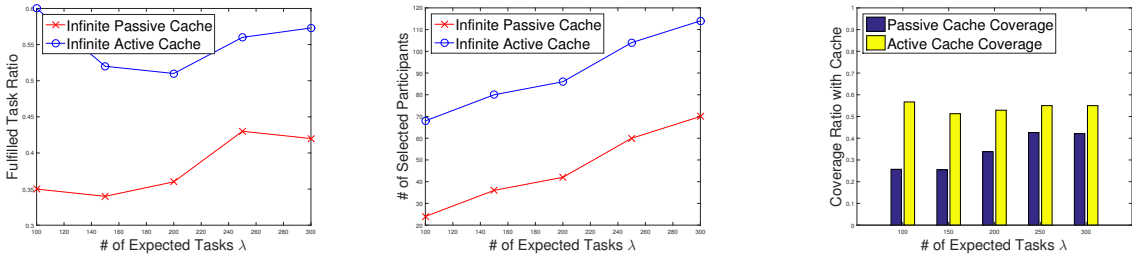
solutions and they are also stable with the increase of number of candidate participants. This shows the advantage of prediction of coverage in participant selection. In addition, the coverage-based algorithm with caching could select fewer participants than the one without caching. That is because when 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. Figure 14(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. Figure 15 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 number of participants.



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

Figure 17: Results 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

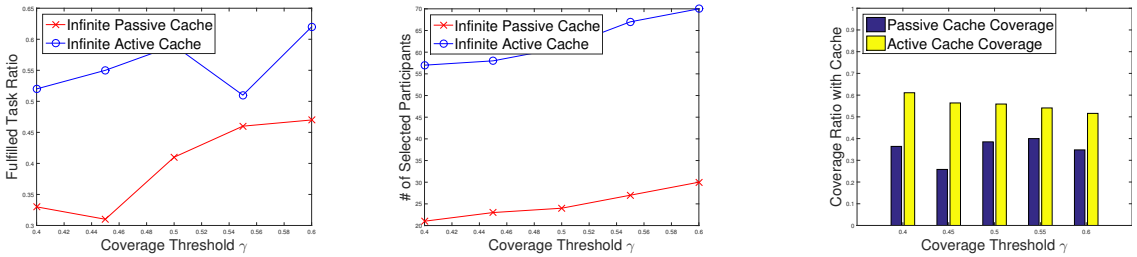
Figure 18: Results in Scenario B when $n = 300$, $\gamma = 0.6$ and $\lambda = 100$ to 300.

Last, we also test different values of the coverage threshold γ (as shown in Figure 16). 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.

4.4.4 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. Figures 17 to 19 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 (Figures 17(a) to 19(a)). Recall that in Scenario B there are Type



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

Figure 19: Results in Scenario B when $n = 300$, $\lambda = 200$ and $\gamma = 0.4$ to 0.6.

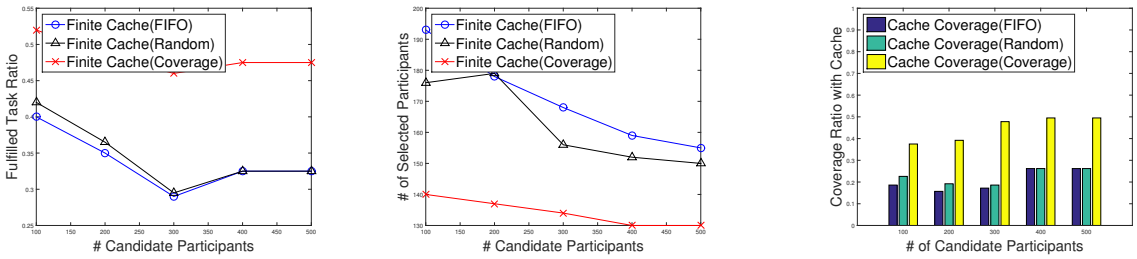
II sensing tasks which cannot be fulfilled by the passive caching. But the passive caching can dynamically assign participants to future coming tasks which leads to higher fulfilled ratios. But the cost of such advantage is more participants selected to perform the sensing tasks (Figures 17(b) to 19(b)).

(2) Similar to Scenario A, as the number of task or the coverage threshold increases, the number of selected participant increases (Figure 18(b) and Figure 19(b)). In addition, the number of selected participants decreases as the number of candidate participants increases (Figures 17(b)).

(3) The fulfilled tasks contributed by active caching could be about $40 \sim 60\%$ of the total fulfilled tasks (Figures 17(c) to 19(c)). This again shows the advantage of active caching due to Type II tasks.

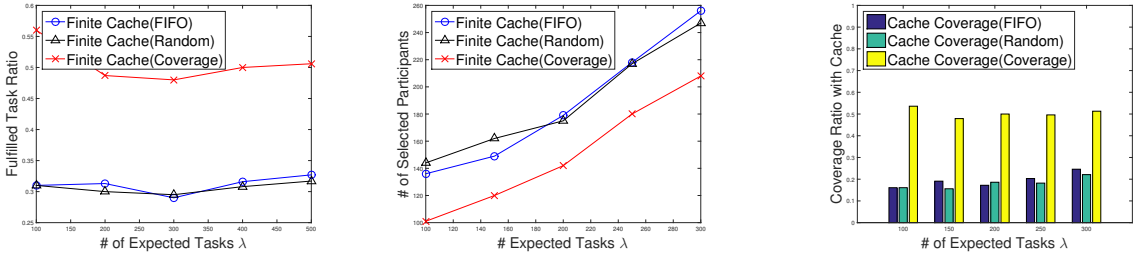
4.4.5 Performance with Different Caching Strategies

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. Figures 20 to 22 show the performance of these three strategies. It is obvious that the strategy based on coverage can achieve



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

Figure 20: Results of different caching strategies when $\lambda = 200$, $\gamma = 0.5$ and $n = 100$ to 500.



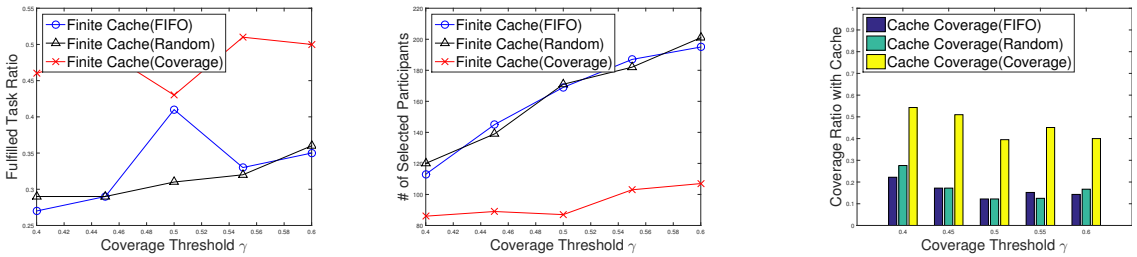
(a) fulfilled task ratio (b) # of selected participants (c) coverage ratio with cache

Figure 21: Results of different caching strategies when $n = 300$, $\gamma = 0.5$ and $\lambda = 100$ to 300.

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

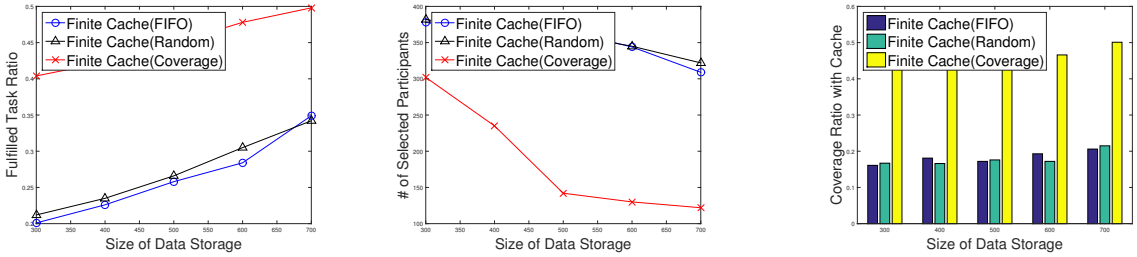
4.4.6 Performance with Different Caching Sizes

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. Figure 23(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 Figure 23(b)). The reason is that more space for sensing records means more chance for effective records being utilized. Figure 23(c) shows that the hitting



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

Figure 22: Results of different caching strategies when $n = 300$, $\lambda = 200$, $\gamma = 0.4$ to 0.6 and $D = 500$.



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

Figure 23: Results of different caching size when $n = 300$, $\lambda = 200$, $\gamma = 0.5$ and $D = 300$ to 700.

rates of different caching schemes are relevantly stable.

4.5 Summary

In this work, 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 sim-

ulations conducted with real-life D4D dataset. We leave further improvements on call prediction as one of our future works.

CHAPTER 5: DATA COLLECTION THROUGH DEVICE-TO-DEVICE COMMUNICATIONS IN MOBILE CROWD SENSING

5.1 Introduction

With the increasing popularity of mobile applications and services for smart devices, we are currently facing the challenges of mobile big data explosion. Based on the most recent Cisco's report [38], mobile data traffic grew 74 percent in 2015 and reached 3.7 exabytes per month at the end of 2015, which was nearly 4,000 times the one in 2005. Cisco also forecasts that mobile data traffic will surpass 30.6 exabytes per month in 2020. Even though smart devices only represent 36 percent of the total mobile devices and connections, they account for 89 percent of the mobile data traffic. The widespread availability of smart devices equipped with a rich set of built-in sensors has also enabled a new sensing paradigm, *mobile crowd sensing* (MCS) [33], for collecting and sharing sensing data from surrounding environment. MCS have been widely used for different sensing applications, such as public safety [10, 39] , traffic planning [12, 14], localization [40, 41], environment monitoring [16, 17], and urban dynamic mining[8, 9]. In the same time, this new sensing paradigm makes the mobile data explosion severer.

The current cellular networks do not have enough capacity to support all of the fast-growing mobile big data from these smart devices and Internet of Things. Different offloading solutions (such as WiFi networks [42, 43] or femtocells [44]) have been

adopted. According to Cisco [38], fifty-one percent of total mobile data traffic was offloaded onto the fixed network through Wi-Fi or femtocell in 2015, and this is the first time offload traffic exceeded cellular traffic. Recently, offloading cellular traffic through opportunistic device-to-device (D2D) communications [45, 46, 47] among mobile phones becomes a new and possible solution. Compared with current WiFi or femtocell solutions, this method uses occasional D2D contact opportunities to deliver data rather than the fixed network infrastructure. The major advantage is low cost and easy to deploy. Han *et al.* [45] study how to select the initial set of mobile users to push the content to all users in the networks via D2D, and their proposed heuristics can improve the delivery efficiency and offload a large fraction of data from the cellular network. Li *et al.* [46] study the problem of multiple mobile data offloading through D2D among different data subscribers under resource constraints. Zhu *et al.* [47] study offloading peer to peer traffic among mobile users with D2D relays. In this work, we focus on offloading data collection for mobile sensing data via D2D relays instead broadcasting traffic from the service provider to all subscriber users (as in [45, 46]) or peer-to-peer traffic between any two users (as in [47]).

In most existing mobile sensing systems [21, 19, 20, 34, 48], the sensing data is collected via cellular networks with the assumption that the size of sensing data is not large. However, with the new types of multimedia sensing (videos, audios, high resolution images, real time streaming, etc.) and increasing number of sensing devices (smart phones, smart watches, smart glasses, smart meters, smart vehicles, RFIDs, etc.), the amount of mobile sensing data grows to a scale that traditional cellular methods may not handle. Wang *et al.* [49] first consider leveraging the delay-

tolerant mechanisms by offloading the data to Bluetooth/WiFi gateways or data-plan users. The major goal for their method is to reduce the energy consumption and data cost of data-plan users. Karaliopoulos *et al.* [50] consider a joint user recruitment problem for both sensing and data collection, where the data collection is done via D2D communications. They formulate the selection of users as a minimum cost set cover problem and propose greedy heuristics to solve it. However, the solution has large time complexity due to the huge search space over all space-time paths across the network, which makes it not suitable for large-scale data collection. In this work, we focus on the data collection phase of mobile data sensing by carefully selecting a few mobile participants as relay nodes to help with data propagation via D2D relays. By doing so, we limit the search space and make our algorithm more efficient. In addition, since we use multiple space-time paths for data collection from the source (in [50] only one space-time path is selected for one source), our method can achieve better delivery ratio too. We also consider the joint problem where the selected participants perform both sensing and data collection.

In summary, in this work, we study how to select a small subset of participants as relaying (or/and sensing) devices so that the data propagation via these D2D relays can achieve certain level of delivery ratio in mobile crowd sensing. We formulate the problem as various optimization problems in Section 5.2. Then we propose simple but efficient solutions in Section 5.3 and Section 5.4 for relay selection and joint relay/sensing selection, respectively. In Section 5.5, we conduct experiments over a real-life mobile trace to confirm the effectiveness of the proposed algorithms. A preliminary version of this work appeared in [51].

5.2 System Model and Problem Statement

5.2.1 System Model

We consider the relay node selection problem for sensing data collection. We assume that a mobile user set $User = \{u_1, u_2, ..., u_n\}$, which includes n mobile users who are willing to participate into sensing and the delivery of the sensing data. This candidate set is assumed very large given the popularity of smartphones. Each mobile users can visit m different locations, denoted as $Location = \{l_1, l_2, ..., l_m\}$. The whole time period is evenly divided into T sequential time slots, thus time $t \in [1, T]$. Each user has her own visiting pattern over both temporal and spacial domains. We use P to denote the visiting probability matrix of all users, which its element $p(u_i, l_j, t)$ (or $p(i, j, t)$ for short) represents the probability of mobile user u_i to make a visit at location l_j during time slot t. There are various methods to estimate the visiting probability of each user based on historical traces, and our proposed solution can use any such existing method. In our simulations, we utilize a simple statistic based method, which is illustrated in Section sec:exp1. We assume that these visiting probabilities are independence to the others for each particular mobile user. We also assume that there is a set of sensing tasks $Task = \{q_1, q_2, ..., q_o\}$, which includes o sensing tasks. Each task i has a tuple of target $(l(q_i), t(q_i))$, which represents the temporal and spacial target of this task. Note here each task only has a single interest point in the temporal and spacial domain, however, it could be easily extended to the case where each task has multiple interest points.

To enable device-to-device communications, we assume that two nodes can discover

each other and transfer sensing data to each other when they both visit the same location within a particular time slot. For each piece of sensing data, it is generated at a source node s (a mobile device which perform the sensing task and generate the data), and needs to be delivered to a sink node d (a mobile device or a static device at certain location). For simplicity, we only focus on the selection of relay nodes for the collection of sensing data to a single sink. However, all proposed methods are general enough to handle multiple sources/sinks. To enhance the delivery probability, we assume that restricted flooding (i.e., Epidemic [52]) is used within the selected relay nodes. Fig. 24 shows an example of data delivery via multi-hop D2D relays. In this figure, dashed curves are trajectories of devices, a circle represents an encounter between two devices, and the device in black indicates that it has a copy of the data. The one marked with "source" is the device performing sensing and generating the data, and the sink is an access point or tower. During the encountering among multiple devices, the data could be transmitted from one device to another. Through this type of multi-hop D2D transmission, sensing data could be delivered to a sink which is not able to directly communicate with the source. Note there we assume the data collection is through only device-to-device communications, while in reality, a hybrid solution (combining D2D and direct communication with cellular tower) could be desired.

5.2.2 Relay Selection Problems

First, we only consider the participant selection for D2D data collection in MCS. For simplicity, we only focus on the selection of relay nodes for the collection of

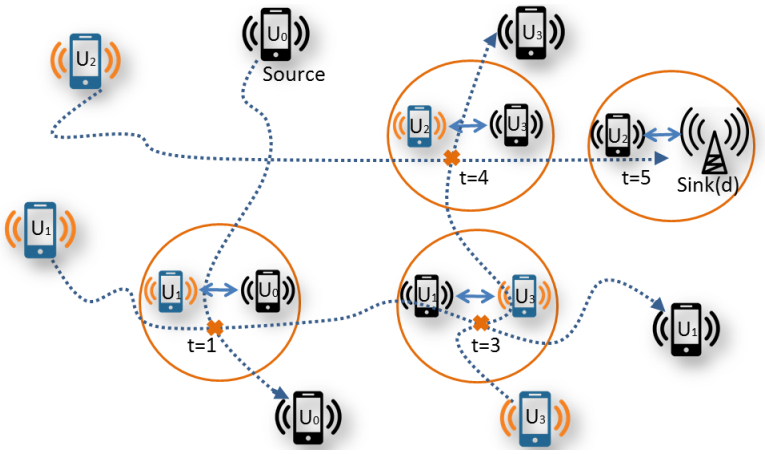


Figure 24: Example of data delivery via multi-hop D2D communications.

a single piece of sensing data. However, the method is general enough to handle multiple data pieces. The key challenging is how to identify a small set of relay nodes from the huge candidate pool $User$ while guarantee certain level of data delivery. This is different with traditional DTN routing, in which relay nodes are dynamically selected during the routing. We can formally define the relay selection problem as the following optimization problem.

Definition 2. Given the volunteering users $User$ (with their historical call and location traces), and the source s (who generates the sensing data) and destination d of the sensing data, *Minimum Relay Problem* is to find a subset $U(s,d)$ of mobile users from $User$ as the relay nodes with the objective to

$$\min |U(s,d)|$$

$$s.t. \;\; p_r(U(s,d),s,d) \geq \gamma.$$

in which $p_r(U(s,d),s,d)$ is denoted as the probability that the sensing data can be delivered to its destination sink and γ represents a threshold of the probability.

Similarly, the optimization problem can be defined as another formulation as well.

Definition 3. Given the volunteering users $User$ (with their historical call and location traces) , and the source s and destination d of the sensing data, *K Relay Problem* is to find a subset $U_{s,d}$ of K mobile user from $User$ as the relay nodes with the objective to

$$\max p_r(U(s,d),s,d)$$

$$s.t.\ \ |U(s,d)| = K.$$

Note that both versions of the problem are computational challenging, since even with perfect predication of visiting patterns this problem can be reduced to a set cover problem which is NP-hard. Therefore, in the next section, we are looking for efficient heuristics to tackle them.

5.2.3 Joint Sensing and Relay Selection Problems

In the problems above, we focus on the selection of participants who will only participate the D2D data collection, by assuming the participants for sensing tasks have been selected and fixed via existing participant selection methods [21, 19, 20, 34, 48]. However, it is very natural to consider the selection of the same group participants for both sensing and data collection purposes. Then, we can define the following joint problems.

Definition 4. Given the volunteering users $User$ (with their historical call and location traces) , and the sensing task q and destination d of the sensing data, *Minimum Sensing & Relay Problem* is to find a subset $U(q,d)$ mobile user from $User$ as the

selected participants with the objective to

$$\min |U(q,d)|$$

$$s.t.\ \ p_s(U(q,d),q,d) \geq \gamma.$$

in which $p_s(U(q,d),q,d)$ is denoted as the probability that the target information can be collected **and** the sensing data can be delivered to the destination sink.

Definition 5. Given the volunteering users $User$ (with their historical call and location traces) , and the sensing task q and destination d of the sensing data, *K Sensing & Relay Problem* is to find a subset $U(q,d)$ of K mobile user from $User$ as the selected participants with the objective to

$$\max p_s(U(q,d),q,d)$$

$$s.t.\ \ |U(q,d)| = K.$$

Note the problems above are defined for a single sensing task q. We can also consider the optimization over the whole sensing task set $Task$, i.e., we select the common set of participants $U(Task,d)$ to perform all sensing tasks in $Task$. The only difference in the definitions is using $\sum_{q_i \in Task} p_s(U(Task,d),q_i,d) \geq \gamma$ as the constraint in the Minimum Sensing & Relay Problem or $\max \sum_{q_i \in Task} p_s(U(Task,d),q_i,d)$ as the optimization goal of the K Sensing & Relay Problem. All of these problems are obviously NP-hard.

5.3 Relay Selection for D2D Collection

Recall that we use a flooding/epidemic strategy to deliver the sensing data via multiple hops among selected relay nodes. The selection criteria for relay nodes may

rely on how to estimate the delivery probability of a particular group of relay nodes. To achieve this, we first introduce a space-time graph based method, then we propose our greedy based algorithm for relay node selection.

5.3.1 Estimation of Delivery Probability via Space-Time Graphs

To capture the evolving characteristics in both spacial and temporal spaces, we adopt the *space-time graph* [53, 54] to model the time-evolving D2D links among selected relay nodes. Let $U(s,d) = \{u_1, \cdots, u_r\}$ is the relay nodes selected for source s and sink d. We can define a space-time graph $\mathcal{G}^{U(s,d)} = (\mathcal{V}, \mathcal{E})$, which is a directed graph defined in both spacial and temporal spaces. Hereafter, we simply use $\mathcal{G}$ to represent $\mathcal{G}^{U(s,d)}$. In $\mathcal{G}$, $T+1$ layers of nodes are defined and each layer has $r+2$ nodes (corresponding to $\{u_0 = s, u_1, \cdots, u_r, u_{r+1} = d\}$), thus the whole vertex set $\mathcal{V} = \{u_j^t | j = 0, \cdots, r+1 \text{ and } t = 0, \cdots, T\}$ and there are $(r+2)(T+1)$ nodes in total. Fig. 25 illustrates the corresponding space-time graph for the network shown in Fig. 24. Two kinds of links (spacial links and temporal links) are added between consecutive layers in the edge set $\mathcal{E}$. A temporal link $\overrightarrow{u_j^{t-1} u_j^t}$ (those horizontal links in Fig. 25) connects the same node u_j across consecutive $(t-1)$th and tth layers, which represents the node carrying the data in the tth time slot. A spacial link $\overrightarrow{u_j^{t-1} u_k^t}$ represents a forwarding possibility from one node u_j to its encountering node u_k in the tth time slot (i.e., u_j encounters u_k in time slot t). By defining the space-time graph $\mathcal{G}$, any communication operation in the time-evolving network can be simulated on this directed graph. E.g., the propagation path in Fig. 24 is highlighted in Fig. 25.

To estimate the delivery probability, we need first define the link probability $p(e)$

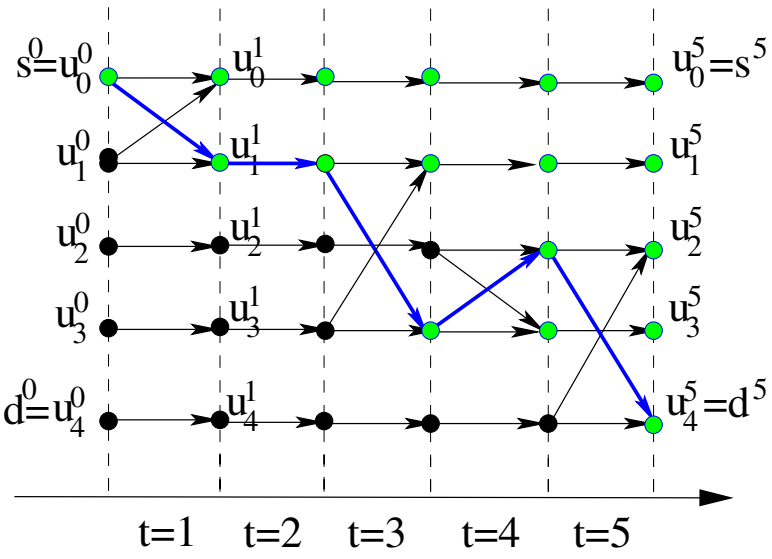


Figure 25: Space time graph: the corresponding space-time graph $\mathcal{G}$ of Fig. 24, where a space-time path from the source s to the sink d is highlighted.

of each link $e \in \mathcal{E}$), i.e. the probability of existing such a link. For each temporal link $\overrightarrow{u_j^{t-1}u_j^t}$, its link probability is set to 1 since a node can always hold the data. For a spacial link $\overrightarrow{u_j^{t-1}u_k^t}$, its link probability is calculated as follows.

$$p(\overrightarrow{u_j^{t-1}u_k^t}) = (1 - \prod_{i=1}^{m}(1 - p(j,i,t)p(k,i,t))) \cdot r(\overrightarrow{u_j^{t-1}u_k^t}),$$

where $1 - \prod_{i=1}^{m}(1 - p(j,i,t)p(k,i,t))$ is the probability that node u_j and u_k are co-located at any location and $r(\overrightarrow{u_j^{t-1}u_k^t})$ is the link reliability (representing the successful transfer over the encounter). If u_k is a location l_k instead of a mobile user, $p(\overrightarrow{u_j^{t-1}u_k^t}) = p(j,k,t) \cdot r(\overrightarrow{u_j^{t-1}u_k^t})$.

We then define the delivery probability of a space-time graph $\mathcal{G}$ is $p^{\mathcal{G}}(s^0, d^T)$ regarding the source s and destination d. It is the probability that a packet sent from node s over the routing topology $\mathcal{G}$ reaches node d under flooding-based routing. Similar definition is used in [55] as broadcast reliability. To efficiently calculate this delivery probability is not an easy job. Actually, it is known that the computation of such

reliability over general graphs is a problem of NP-hard [56]. Fortunately, the nice loop-free property of our space-time graph model allows us to compute the reliability very efficiently with a dynamic programming (DP) algorithm [55]. Basically, for any node u_i^t in $\mathcal{G}$, its delivery probability from the source node s can be calculated as follows:

$$p^{\mathcal{G}}(s^0, u_i^t) = 1 - \prod_{\overrightarrow{u_j^{t-1}u_i^t} \in \mathcal{G}} (1 - p^{\mathcal{G}}(s^0, u_j^{t-1}) p(\overrightarrow{u_j^{t-1}u_i^t})).$$

Given the structure $\mathcal{G}$ defined by r relay nodes, starting from a source node, the dynamic programming algorithm can compute the delivery ratio of all other nodes within time of $O(rT(\log(rT) + r))$. Notice that the time complexity of DP algorithm is the same with that of Dijkstra's algorithm. Given the relay node set $U(s, d)$ for source s and sink d, we can estimate the delivery probability based on the space-time graph $\mathcal{G}$ as follows: $p_r(U(s, d), s, d) = p^{\mathcal{G}}(s^0, d^T)$.

5.3.2 Relay Selection Algorithm

Then the relay selection algorithm is quite straightforward. In each step, we greedily select the user u_i which leads to maximal improvement of $p_r(U(s, d), s, d)$ into $U(s, d)$. Repeat this until either the delivery probability reaches the threshold γ for minimum relay problem or $U(s, d)$ has K users for k relay problem. However, there is still a starting problem, since initially when $U(s, d)$ is empty or just with a few users the space time graph $\mathcal{G}^{U(s,d)}$ may not be connected at all (i.e., $p_r(U(s, d), s, d) = 0$). In this case, adding any single user may not improve the delivery probability. Therefore, instead of considering improvement of $p_r(U(s, d), s, d)$, we simply pick the user who is the most active (in term of visited locations). Detailed algorithm is given as

Algorithm 5 Relay Selection Algorithm

Input: potential user set $User$, visiting probability matrix P of all users in $User$, the source s and the sink d.
Output: selected relay nodes $U(s,d)$.

1: $U(s,d) = \emptyset$
2: **while** $\mathcal{G}^{U(s,d)}$ is not connected **do**
3: Choose the most active user and add it into $U(s,d)$
4: **end while**
5: **while** $|U(s,d)| < K$ or $p_r(U(s,d),s,d) < \gamma$ (for K *relay problem* or *minimum relay problem*, respectively) **do**
6: **for all** $u_i \in User$ and $\notin U(s,d)$ **do**
7: Calculate the improvement of $p_r(U(s,d),s,d)$ by adding u_i in to $U(s,d)$ (Section 5.3)
8: **end for**
9: Select the user u_i with the largest reliability improvement and add it into $U(s,d)$
10: **end while**
11: **return** $U(s,d)$

Algorithm 5.

Next we consider the time complexity of Algorithm 5. Since the complexity of initial step to form a connected space time graph (lines $2-3$) is much smaller than the complexity of relay selection step (lines $4-7$), we only focus on the latter. First, the while loop will be performed K rounds in K relay problem and n rounds in minimum relay problem since in the worst case we need to select all participants to achieve the threshold. Second, the for loop is bounded by n rounds. Third, the complexity of DP algorithm for the estimation of delivery probability is $O(rT(log(rT)+r))$ and r is the size of selected relay group which is bounded by K or n in K relay problem and minimum relay problem. Therefore, the time complexity of Algorithm 5 is bounded by $O(nKT(log(KT)+K))$ or $O(n^2T(log(nT)+n))$ for K relay problem or minimum relay problem, respectively.

5.4 Joint Sensing and Relay Selection

When we consider the joint sensing and relay selection problems (defined in Definition 4 and Definition 5), we have to first estimate the sensing capability of each participants and then integrate it into the participant selection procedure.

5.4.1 Estimation of Sensing & Delivery Probability

Recall that each task has a target tuple of location and time. Therefore, given a specific task q_i and a set of selected participant $U(q_i, d)$, the probability that this task can be performed by one participant $u_j \in U(q_i, d)$ can be obtained. Basically, the probability of u_j making a visit to location $l(q_i)$ at time $t(q_i)$ is $p(u_j, l(q_i), t(q_i))$.

Here, we assume that the probability of sensing and data delivery is independent to each another. Therefore, the probability of sensing and delivery of a sensing task q_i with a particular source can be calculated by multiply the sensing probability (at time $t(q_i)$) with the delivery probability (from time $t(q_i)$ to T over the space-time graph). Thus, given a sensing task q_i and a selected participant $u_j \in U(q_i, d)$ as the source (the sensing performer), the sensing & delivery probability from this participant is

$$p_s(u_j, U(q_i, d), q_i, d) = p(u_j, l(q_i), t(q_i)) \cdot p^{\mathcal{G}}(u_j^{t(q_i)}, d^T)$$

Then overall sensing & relay probability of task q_i by $U(q_i, d)$ is:

$$p_s(U(q_i, d), q_i, d) = 1 - \prod_{u_j \in U(q_i, d)} (1 - p_s(u_j, U(q_i, d), q_i, d)).$$

Algorithm 6 Sensing and Relay Selection Algorithm

Input: potential user set $User$, visiting probability matrix P of all users in $User$, the sensing task q and the sink d.
Output: selected relay nodes $U(q,d)$.

$U(q,d) = \emptyset$
while $\mathcal{G}^{U(q,d)}$ is not connected **do**
  Choose the most active user and add it into $U(q,d)$
end while
while $|U(q,d)| < K$ or $p_s(U(q,d),q,d) < \gamma$ (for *K sensing & relay problem* or *minimum sensing & relay problem*, respectively) **do**
  for all $u_i \in User$ and $\notin U(q,d)$ **do**
    Calculate the improvement of $p_s(U(q,d),q,d)$ by adding u_i in to $U(q,d)$ (Section 5.4)
  end for
  Select the user u_i with the largest sensing and relay improvement and add it into $U(q,d)$
end while
return $U(q,d)$

5.4.2 Sensing and Relay Selection Algorithm

The participant selection algorithm basically is still the same. The only difference is now the joint sensing and relay probability is considered instead of relay probability. Algorithm 6 shows the detailed algorithm. If we consider the selection over the whole sensing task set $Task$, the improvement of sensing and relay probability should be over the summation of all sensing tasks, i.e. the improvement of $\sum_{q_i \in Task} p_s(U(Task,d),q_i,d)$. Since Algorithm 6 is similar to the one for relay selection, the time complexity of this algorithm is bounded by $O(nKT(log(KT)+K))$ or $O(n^2T(log(nT)+n))$ for K relay problem or minimum relay problem as well.

5.5 Experiments over D4D Dataset

To evaluate the proposed algorithms, we conduct extensive simulations over D4D dataset [24]. To make comparisons, we also implement two simple heuristics: *random*

selection and *activity-based selection*. Random selection randomly chooses a user at each step until the algorithm ends, while activity-based selection greedily chooses a user who is most active (visiting most locations) at each step. In our simulations, we use the real delivery ratio and the number of selected users as the metrics of measurement.

5.5.1 Experiment Settings

We select 20 most popular towers, i.e. the towers with largest number of associated records, to implement our simulations. Thus, $m = 20$. We choose a 100 candidate user set as $User$, i.e., $n = 100$. For each sensing task, we randomly generate its request at one of the towers and at one of the time slots from 1 to T. We assume that the whole sensing period T is one week and treat one hour as the smallest time unit. For each data collection task, we randomly select one location as the sink. If two users make phone calls associated with same tower at same time, we assume that they are close to each other and could transfer data between them via D2D links. For simplicity, we set the link reliability as 0.5, i.e., the successful transferring over a pair of nodes is 50% during their encountering.

To calculate the probability $p(u_i, l_j, t)$ of a particular user u_i visiting a location l_j at certain time t, we have to leverage the knowledge from the historical call traces. Here, we assume that for each user we have multiple rounds of call traces (e.g., X weeks), and each round of data denoted as D_i, $i = 1, \cdots, W$. Let $c_x(u_i, l_j, t)$ indicate whether u_i made one or more phone call at l_j and t in D_x (1 if it made, 0 otherwise).

Then we simply estimate the visiting probability as follows,

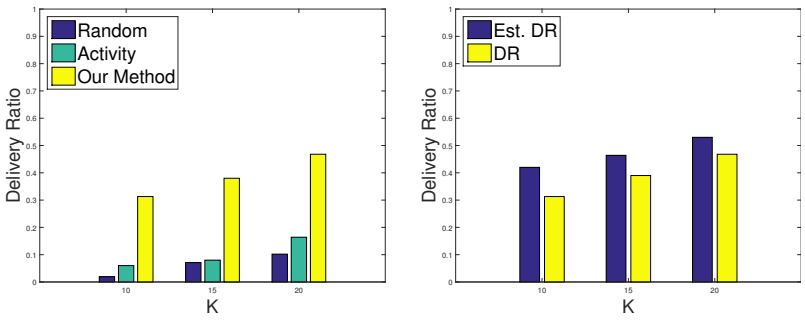
$$p(u_i, l_j, t) = \frac{\sum_{x=1}^{W} c_x(u_i, l_j, t)}{W}.$$

Instead of this simple model, we can also consider Markov model [22] or Poisson process [19].

5.5.2 Experiments on D2D Data Collection

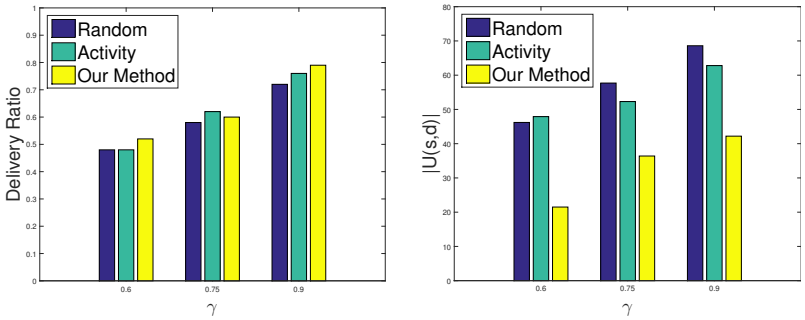
We first test our proposed algorithm (Algorithm 5) for D2D data collection of sensed data (K relay problem and minimum relay problem). For each data collection task, we randomly select a mobile user as the data source. For each set of experiments, we test 15 tasks and 100 rounds per tasks. The average performances over $1,500$ rounds are reported.

In the first set of simulations, we consider the K relay problem. We vary the number of selected relay nodes from 10 to 20. Fig. 26(a) shows the delivery rate achieved by each algorithm. It is clear that our proposed algorithm achieves the highest delivery ratio among the three algorithms when the number of selected relay nodes are the same. In addition, we can find that the delivery ratio of all the three algorithms increase as the number of selected relay nodes increase. This is obvious since more selected relay nodes provide more possible routes for the data to reach the sink node. Fig. 26(b) shows the comparison between expected delivery ratio and real delivery ratio of our proposed algorithm. The real delivery ratio is always lower than the expected one since the estimation is based on the historical records. Although it is not 100 percent accurate, the expected delivery ratio still provides us the guidance



(a) delivery ratio (DR) (b) estimated DR vs DR

Figure 26: Results for K relay problem where $K = 10$, 15 or 20.

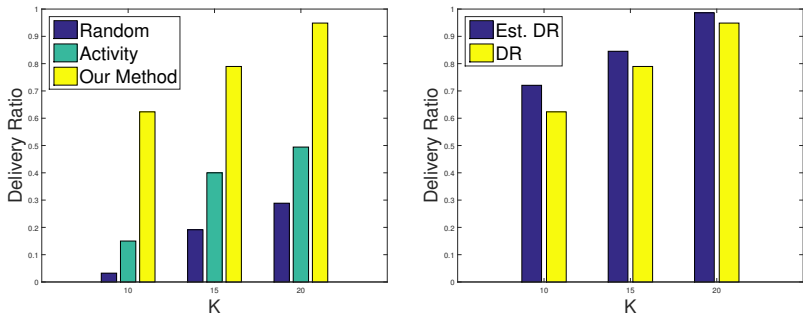


(a) delivery ratio (b) number of relay nodes

Figure 27: Results for minimum relay problem where $\gamma = 0.6$, 0.75 or 0.9.

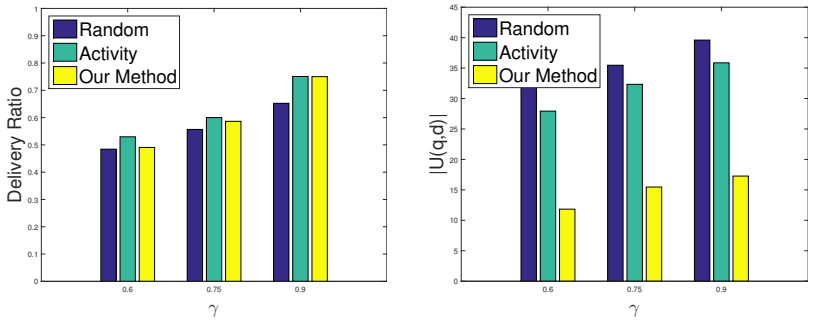
to pick the relay nodes.

In this set of simulations, we evaluate the performance of algorithms over minimum relay problem. Here we vary the delivery ratio threshold γ from 0.6 to 0.9. Fig. 27(a) shows that the delivery ratios of the three algorithms are similar to each others. Recall that all algorithms will continue add new relay nodes until the estimated delivery probability reach the threshold. Since the threshold is the same for the three algorithms, the overall delivery ratios are similar. However, in Fig. 27(b), we can see that the number of selected relay nodes of our algorithm is much fewer than those of the other algorithms when achieving similar level of delivery ratios. This confirms that our proposed algorithm is more efficient than the other two simple heuristics.



(a) delivery ratio (DR) (b) estimated DR vs DR

Figure 28: Results for K sensing & relay problem with a single sensing task where $K = 10$, 15 or 20.

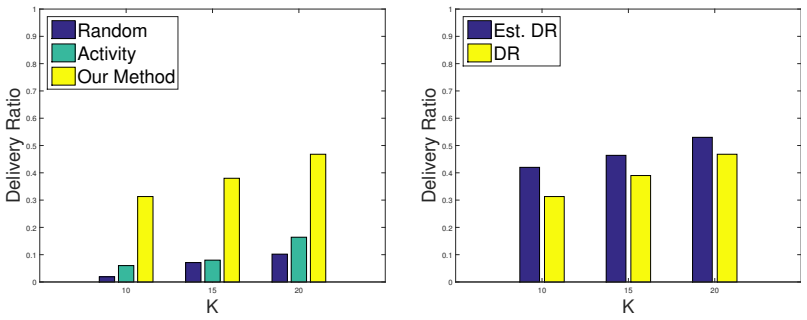


(a) delivery ratio (b) number of selected nodes

Figure 29: Results for minimum sensing & relay problem with a single sensing task where $\gamma = 0.6$, 0.75 or 0.9.

5.5.3 Experiments on Joint Sensing and D2D Relay

We also test our proposed algorithm (Algorithm 6) for joint sensing and relay problems. Here we consider two cases: (1) the optimization of participant selection is done per task based; and (2) the optimization of participant selection is done for multiple tasks. For each set of experiments for the first case, we test 15 different tasks and 100 rounds per task. For each set of experiments for the second case, we test several task sets with various numbers of tasks and perform participant selection over 100 round per task set. For each sensing task, its interested location is generated



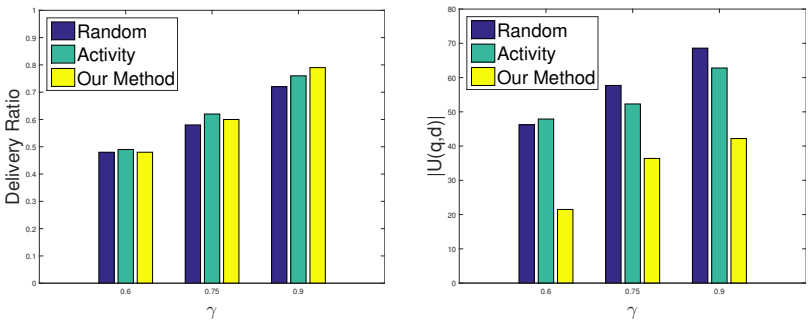
(a) delivery ratio (DR) (b) estimated DR vs DR

Figure 30: Results for K sensing & relay problem with 5 sensing tasks where $K = 10$, 15 or 20.

randomly.

For K sensing & relay selection, we vary the number of selected relay nodes H from 10 to 20. Fig. 28(a) shows the delivery rate achieved by each algorithm. Similar to the results of the experiments on K relay selection, our proposed algorithm achieves the highest delivery ratio among the three algorithms when the number of selected nodes are the same. The delivery ratio increases as the number of selected participants increases. Fig. 28(b) also shows that there are still differences between the expected delivery ratio and real delivery ratio of our proposed algorithm since the estimation is based on the historical records. In the simulations of minimum sensing & relay selection, we vary the delivery ratio threshold γ from 0.6 to 0.9. Fig. 29(a) shows that the three different algorithms achieve similar delivery ratio under the same delivery ratio threshold. However, our proposed algorithm selects fewer participants than those of the other two algorithms (Fig. 29(b)).

We then test our proposed algorithm in multi-task scenario where the selection of participants is optimized for the whole task set $Task$. Firstly, we test our algorithm over task sets with 5 sensing tasks. Fig. 30 and Fig. 31 show the performance of K



(a) delivery ratio (b) number of selected nodes

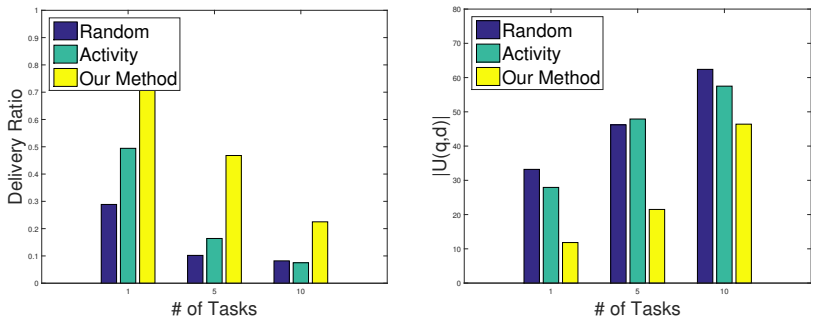
Figure 31: Results for minimum sensing & relay problem with 5 sensing tasks where $\gamma = 0.6$, 0.75 or 0.9.

sensing & relay problem and minimum sensing & relay problem respectively. The trends are similar to those in the single-task cases.

We also vary the number of sensing tasks in $Task$ from 1 to 10 to evaluate the performance over different size of task set. Fig. 32(a) shows the changing of delivery ratio when we fix the K to 20 and γ to 0.6 for K sensing & relay problem. We find that the delivery ratio drops when the number of tasks increases. Fig. 32(b) shows the number of selected participants increases along the increasing of tasks when we fixed γ in minimum sensing & relay problem. Both of the trends above are reasonable since more tasks usually take more participants to perform and relay the data. One one hand, it needs more people to perform the sensing tasks. On the other hand, it also needs more possible path to relay the sensing data to the destination.

5.6 Summary

In this work, we investigate the feasibility of collecting data packets from mobile devices in mobile sensing through device-to-device communications, by carefully selecting the subset of participant devices. We formulate the problem as several op-



(a) K sensing & relay (b) minimum sensing & relay

Figure 32: Results for K sensing & relay problem ($K = 10$) and minimum sensing & relay problem where the number of sensing tasks $o = 1$, 5 or 10.

timization problems (K relay selection, minimum relay selection, K sensing & relay selection, and minimum sensing & relay selection) and propose simple greedy algorithms to solve them. The proposed algorithms use historical information to obtain the estimated sensing and delivery probability of a given participant set and greedily select the participant based on this estimated probability. Our experiments over the real-life D4D mobile traces confirm the effectiveness of the proposed algorithms.

As for future works, we plan to continue our study on data collection in mobile sensing along the following directions: (1) hybrid data collection schemes which combine D2D and direct communications; (2) implementation of the proposed methods over a real testbed with mobile devices and experiments with real world sensing tasks; and (3) modeling of energy consumption in mobile sensing and designing new energy efficient solutions.

CHAPTER 6: MULTI-EXPERTISE AWARE PARTICIPANT SELECTION IN MOBILE CROWD SENSING VIA ONLINE LEARNING

6.1 Introduction

Smart mobile devices with embedded sensors have been developing rapidly, which enables a new sensing paradigm *mobile crowding sensing* (MCS). By leveraging large number of mobile device owners, MCS can collect tremendous data without additional infrastructure cost [7, 33]. There are also advantages such as providing better temporal and spacial coverages, observing unpredictable events, and integrating human intelligence compared with traditional wireless sensor networks. Several MCS applications have been used in real world such as dynamic behavior mining [8, 9], environment monitoring [15, 16, 17], traffic planning [11, 12, 13, 14] and public safety monitoring [10]. MCS brings convenient, flexible and efficient solutions for large-scale heterogeneous sensing tasks. However, it also brings new challenges into the system design due to its large-scale tasks and participants. One of the most challenging problems is the *participant selection problem*.

It is challenging to select participants from a large candidate pool to perform heterogeneous sensing tasks. There are different constraints needed to be taken into consideration such as coverage [63, 34, 64], cost [18, 21, 65] and quality [66, 67]. In most existing participant selection methods, the MCS platform usually needs the participant information, such as their mobility patterns and sensing qualities, to se-

lect the appropriate participants for certain tasks more efficiently in many real world scenarios. However, it is difficult to for MCS system to retrieve such participant information due to lack of history records or privacy concerns [68, 69, 70]. Therefore, how to self learn the valued information of the participants and utilize it to improve the performance of the selection becomes a critical issue in large-scale MCS systems.

There are several studies begin to focus on such type of learning problems [71, 72, 73] in MCS. However, [71] only considers learning sensing quality for homogeneous sensing tasks, where the sensing quality is modeled as random normalized variables. [72] focuses on the online labeling problem in which the true label is unknown. It only considers homogeneous sensing tasks with one-dimension metric. [73] consider heterogeneous tasks but learn the participant's expertise based on semantic analysis. Instead, the real heterogeneous tasks could have different temporal and spatial requirements, and the participant's capability could also include the data collection performance (such as time delay caused by different access methods or availabilities). In other words, the participant may have multi-expertise to perform sensing tasks. Therefore, in this work, we study the learning method for a dynamic participant selection problem with multi-expertise (both sensing probability and time delay).

To address this issue, we formulate and investigate a dynamic *participant selection problem* (PSP) in this work. In our MCS model, the selection mechanism selects the participants for heterogeneous sensing tasks on both spatial and temporal domain. The selected participant has unknown probability to perform the assigned sensing tasks. After performing sensing tasks, the participant may utilize different ways to upload collected sensing data, such as WiFi, Device-to-Device (D2D) or cellular

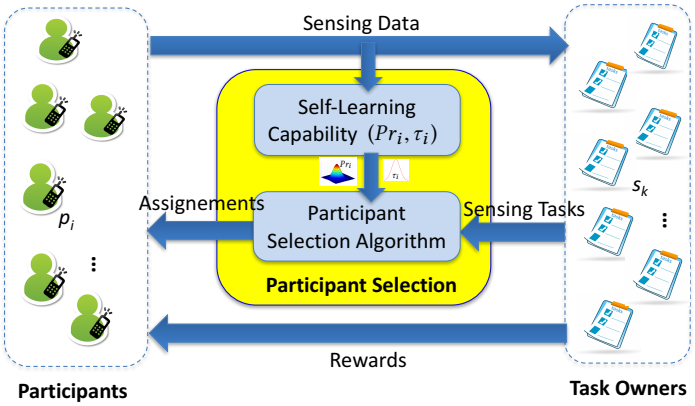


Figure 33: The framework of the proposed online learning for participant selection in MCS.

networks. Different uploading methods may lead to different delay time, which is still unknown by the selection mechanism. Having the historical records of participant performing tasks, we propose online self-learning algorithms to solve this participant selection problem by combinational multi-armed bandit (CMAB) concepts [74]. We show that our proposed methods can achieve both bounded performance in each round of PSP and bounded regret compared with optimal solution over time. We have conducted extensive simulations over a real-life mobile dataset (D4D data set [24]) to evaluate the proposed algorithms in different MCS settings. Our results show that the proposed methods can achieve stable task coverages while the regret is bounded compared with optimal solution. The main contributions of this work are summarized as below.

- We propose a new MCS system model to capture the *multi-expertise* of each participant in both sensing and data uploading stages for heterogeneous sensing tasks on both spatial and temporal domain.

- We formulate a dynamic *participant selection problem* (PSP) as a combinational multi-armed bandit problem, prove its NP-hardness, and propose an online learning algorithm to solve this problem.

- We provide theoretical performance guarantees of the proposed method: an $(1 - 1/e)/2$-approximation for PSP in each round and a bounded regret over time.

- We implement our solutions over a real-world mobile dataset and conduct extensive simulations to evaluate the proposed methods against both existing solutions and optimal solutions.

The rest of this work is organized as follows. We first introduce our MCS system model and the corresponding participant selection problem in Section 6.2. Then we provide the detailed design of proposed online-learning participant selection algorithms in Section 6.3. Section 6.4 presents our simulation results over a real-life mobile tracing dataset. Section **??** reviews the related works. Finally, Section 6.5 concludes this work.

6.2 System Models and Problem Statement

6.2.1 Crowdsensing Model and Assumptions

As shown in Figure 33, there are three main components in our proposed mobile crowd sensing (MCS) model: a set of *sensing tasks*, a number of *mobile participants*, and a *participant recruitment mechanism*. The sensing tasks are generated by crowd sensing applications continuously and then sent to the participant recruitment

mechanism. The mobile participants are mobile users, who are willing to participate in performing mobile sensing tasks. The participants must register to the MCS platform so that they could be assigned with sensing tasks. The participant recruitment mechanism is a centralized mechanism in MCS system in which the decisions of participant selection and task assignments are made. In our design, there are two sub-components inside the participant selection mechanism: *self-learning capability* and *participant selection algorithm*. The self-learning capability analyzes the performance of each participant to learn its capability, while the participant selection algorithm selects the participants for each task in each round. We assume that task assignments from selection algorithm can be sent to selected participants via cellular networks instantly at any time. However, the collected sensing data take longer time to the MCS platform (or MCS applications) via diverse methods (such as WiFi offloading or D2D relays).

6.2.1.1 Sensing Tasks

There is a set of m heterogeneous sensing tasks, denoted by $\mathbb{S} = \{s_1, s_2, ..., s_m\}$. Here the tasks are time sequential, i.e. for each task pair s_i and s_j, if $i < j$, then s_i comes earlier than s_j. Each sensing task has its own target information in both spatial and temporal domains. In this work, we assume that there is a finite set of r locations and a finite set of T time slots, denoted by $\mathbb{L} = \{l_1, l_2, ..., l_r\}$ and $\mathbb{T} = \{1, 2, ..., T\}$ respectively. These tasks are all generated by the MCS applications within time period from 1 to T repeatedly.

Each sensing task s_k specifics a tuple $< l(s_k), t(s_k) >$ as its target information[5].

[5]Note that here we assume that each sensing task only has one interested point in both temporal

Here $l(s_k)$ and $t(s_k)$ is the interested location and time slot of sensing task s_k, respectively. I.e., a sensing task $s_k(l(s_k), t(s_k))$ requires the participants to collect sensing information from location $l(s_k)$ at time $t(s_k)$ and upload such information to the platform.

Here, we assume that each sensing task has two different stages to be performed, *sensing stage* and *uploading stage*. In *sensing stage*, the target sensing information of a task needs to be collected by selected participant(s) at the target location within the target time slot. Successfully collecting such information is only half of the task, in *uploading stage*, the participant(s) who collect the target sensing information need to upload the collected data to the platform. The collection of sensing data can be done by WiFi offloading [75] or D2D relays [76, 77]. Only when the MCS platform receives the requested sensing information from no fewer than one participant, the sensing task is marked as successfully performed.

6.2.1.2 Mobile Participants

There is a set of n participants, who are willing to participate into performing sensing tasks, denoted by $\mathbb{P} = \{p_1, p_2, ..., p_n\}$. These participants register into the MSC system so that they could be considered as candidates. Each participant p_i has her own *capability* to perform sensing tasks represented by a tuple $< \boldsymbol{Pr}_i, \tau_i >$. Here, we consider two types of expertise for each participant to illustrate how to handle multiple expertise during participant selection.

First, $\boldsymbol{Pr_i}$ is a $r \times T$ matrix represents the expertise of participant p_i in term

and spacial domains, but it is easy to relax such an assumption to handle complex sensing tasks with multiple interested points.

of mobility pattern over both spacial and temporal domains (i.e., each participant p_i has her own probability $Pr_i(l_j, t_s)$ that i will arrive at location l_j at time t_s in *sensing stage*). We call such probability *sensing probability* and the matrix $\boldsymbol{Pr_i} = \{Pr_i(l_j, t_s)\}, (l_j \in \mathbb{L}, t_s \in \mathbb{T})$. Therefore, if we use D as a random variable to indicate whether participant p_i arrives location l_j at time t_s (if yes, $D = 1$, else $D = 0$), then D is subject to a Bernoulli distribution $B(1, Pr_i(l_j, t_s))$.

Second, τ_i represents the expertise of participant p_i in term of communication capability (i.e., the expected time for participant i to upload the collected sensing data during *uploading stage* after she is selected and succeeds in *sensing stage*). We call such time *time delay*. Here we use a random variable Γ to represent this time delay of participant p_i with any given sensing task. Then we assume that Γ is subject to a normal distribution with the mean of τ_i, i.e. τ_i is the expected time delay of p_i.

This *capability* of $< \boldsymbol{Pr_i}, \tau_i >$ is critical and necessary knowledge for participant selection. The *capability* of each participant is unknown to the MCS platform (i.e., recruitment mechanism). The platform can only know the values of those random variable outputs after each time the selected participants performing the sensing tasks. We assume that the participants are trusted and truthful so that they will report if they do not perform the sensing tasks. Participants will be rewarded if they perform sensing tasks. We assume that each participant p_i has her own price r_i for performing each task.

6.2.2 Estimation of Participant's Capability

As discussed above, there are two unknown attributes of participant capability in this work, *sensing probability* and *time delay*. We now discuss how to estimate them.

6.2.2.1 Sensing Probability

Since we cannot foreknow when and where a participant will arrive the task location during the real crowding sensing period T (e.g., one week), we have to leverage knowledge from the historical performances. Here, we assume that for each user we have multiple rounds of performance traces (e.g., K weeks), and each round of data denoted as R_i, $i = 1, \cdots, K$. Let $c_k(p_i, l_j, t_s)$ indicate whether p_i successfully visit l_j at t_s in R_i (1 if it visits, 0 otherwise). Then we simply estimate the sensing probability as follow,

$$\tilde{Pr}_i(l_j, t_s) = \frac{\sum_{k=1}^{K} c_k(p_i, l_j, t_s)}{K}. \tag{9}$$

6.2.2.2 Time Delay

In this work, sensing tasks are assigned to participants at any time by sending a task assignment. After the selected participants perform the sensing tasks, they have to upload the collected data back to the MCS platform so that they can receive their rewards. However, there are time delays before the participants complete uploading their data to the platform. One one hand, it takes time to upload the data with various bandwidths and sizes of data. On the other hand, participants may not want to upload the data through cellular networks, but wait for some free or cheap networks access (such as WiFi access points or D2D replays). We define the time

delay as the time from the ending of the sensing stage until the finish of the data uploading. Different participants may have different network access patterns thus each participant p_i has her own delay time τ_i on MCS tasks. Assume that each participant p_i is associated with a set of random variables $T_{i,\gamma}$. Variable $T_{i,\gamma}$ indicates the random delay time of p_i in its γ-th selection. These variables are independent and objected to a distribution with expectation τ_i.

Then we can use the following method to estimate τ_i:

$$\tilde{\tau_i} = \frac{\sum_{k=1}^{K} c_k(p_i)\tau_{i,k}}{K}, \tag{10}$$

where $c_k(p_i)$ indicates whether p_i successfully uploads the data, and $\tau_{i,k}$ is the delay time of p_i to perform the task in k-th round. This time delay represents the computing capability and encountering frequency to WiFi APs or D2D relays of each participant, which may vary among different tasks. The participants with shorter delay attribute has bigger probability to competing the uploading collected data faster than the ones with longer delay attribute.

Note that we model two expertise Pr_i and τ_i quite differently, Pr_i depends on the target location and time while τ_i did not. One of the advantages of the proposed framework for multi-expertise solution is allowing separately model different expertise of participants.

6.2.3 Participant Selection Problem

In our proposed MCS system, MCS tasks arrive at the recruitment mechanism at each discrete round $k = 0, 1, ..., m$. For simplicity, we assume that each task arrived

per round. However, a simple extension can handle multiple tasks per round. Note here a round is much larger than a time slot (which is used for sensing task, where each task has a target time within $[0, T]$). For each task s_k, the selection mechanism selects a subset P_k of $\mathbb{P}$ to perform it. Here we denote $\boldsymbol{\theta}_k = (\theta_{1,k}, \theta_{2,k}, ..., \theta_{n,k})$ as an indicator vector, in which $\theta_{i,k} \in \{0, 1\}$ indicates whether participant p_i is selected for task s_k. If so, $\theta_{i,k} = 1$; else $\theta_{i,k} = 0$.

Denoted by $U(p_i, l_j, t_s)$, the expected utility of selecting a particular participant p_i to perform task with requirement (l_j, t_s) can be defined as follows:

$$U(p_i, l_j, t_s) = Pr_i(l_j, t_s)\phi(\tau_i), \tag{11}$$

where $\phi(\tau_i)$ is a non-negative decay function that represents the decay of the value of the collected data in term of time delay, whose value is within $[0, 1]$. Recall that τ_i is the expected time delay needed by participant p_i for uploading her data after performing sensing task. The value of $U(p_i, l_j, t_s)$ is in $[0, 1]$ as well.

Since the true values of Pr_i and τ_i are unknown and the estimations of them are updated along the time by our learning algorithm, we have an empirical $\tilde{Pr}_i$ and $\tilde{\tau}_i$ for each p_i at any time. Then we have the empirical utility $\tilde{U}(p_i, l_j, t_s)$ at any time as

$$\tilde{U}(p_i, l_j, t_s) = \tilde{Pr}_i(l_j, t_s)\phi(\tilde{\tau}_i). \tag{12}$$

Then the overall expected utility of a given participant subset P_k on a given task $s_k(l(s_k), t(s_k))$ as:

$$\mathbf{U}(P_k, l(s_k), t(s_k)) = 1 - \Pi_{p_i \in P_k}(1 - U(p_i, l(s_k), t(s_k))). \tag{13}$$

Finally, the participant selection problem we aim to solve in this work can be defined.

Definition 6. Given the volunteering participants $\mathbb{P}$ and the crowd sensing task set $\mathbb{S}$, for each task $s_k(l(s_k), t(s_k)) \in \mathbb{S}$, the **participant selection problem (PSP)** tries to select a participant set P_k (and its corresponding $\boldsymbol{\theta}_k$) with the objective of maximizing the overall expected utility:

$$\begin{aligned} &\max_{P_k \subseteq \mathbb{P}} \mathbf{U}(P_k, l(s_k), t(s_k)) \\ s.t. \quad &\sum_{i \in [1,n]} r_i \theta_{i,k} \leq B, \end{aligned} \tag{14}$$

in which the constraint restricts that the total cost of each task must be smaller that a predefined budget B. Let $P_k^*(l(s_k), t(s_k))$ and $\boldsymbol{\theta}_k^*$ be the optimal solution.

Theorem 2. The participant selection problem is NP-hard even if the $\boldsymbol{Pr}$ and $\boldsymbol{\tau}$ is known.

Proof. This can be proved from the reduction of the 0-1 knapsack problem. In 0-1 knapsack problem, there is a set of items, each of them i has a weight w_i and value v_i. The objective of the problem is to maximize the sum of the values of the items while the sum of the weights is less or equal than the given weight capacity W. Our problem is a special instance of 0-1 knapsack problem, in which participants are the items. The value and the weight of each item is replaced by the utility and reward of each participant respectively. The only difference is that our problem goal is not the sum of the values. However, these two goals are equivalent since each individual utility of participant (i.e., $U(p_i, l(s_k), t(s_k))$) is independent. Since 0-1 knapsack problem is an

Algorithm 7 Online Algorithm for Participant Selection

Input: participant pool $\mathbb{P}$, task set $\mathbb{S}$, and price r_i of each p_i
Output: P_k for every task $s_k(l(s_k), t(s_k)) \in \mathbb{S}$

1: Initialization: set $\tilde{\boldsymbol{P}}r_i$ and $\tilde{\boldsymbol{\tau}}_i$ to some values (such as 1s)
2: **for** $k = 1$ to m **do**
3: {*Participant Selection*: Line 4-5}
4: Run Algorithm 2 and Algorithm 3, select and output the P_k with higher utility
5: Update $N_k(p_i, l(s_k), t(s_k))$ if $p_i \in P_k$
6: {*Self-Learning Capability*: Line 7-14}
7: **if** sensing data from p_i is uploaded **then**
8: update $\tilde{Pr}_i$ with a success and $\tilde{\tau}_i$ with measured delay
9: **else if** p_i visits $l(s_k)$ at $t(s_k)$ **then**
10: update $\tilde{Pr}_i$ with a success and $\tilde{\tau}_i$ with maximal delay
11: **else**
12: update $\tilde{Pr}_i$ with a failure
13: **end if**
14: Update $\tilde{U}(p_i, l(s_k), t(s_k))$ with Pr_i and τ_i
15: Set $\bar{U}(p_i, l(s_k), t(s_k)) = \tilde{U}(p_i, l(s_k), t(s_k)) + \sqrt{\frac{3 \ln k}{2N_k(p_i, l(s_k), t(s_k))}}$
16: **end for**

known NP-hard problem thus our problem is NP-hard as well. □

6.3 Participant Selection via Online Learning

We provide the design of our online learning algorithm in this section. Utilizing the proposed algorithm, we estimate $\boldsymbol{Pr_i}$ and τ_i of each participant in each round and make the online participant selection for performing sensing tasks.

6.3.1 Online Algorithm

As defined above, the PSP could be categorized as a *combinational multi-armed bandits* (CMAB) [74]. In a CMAB problem, each arm is associated with a set of random variables which represent the reward of this arm. The reward of each arm is independent to each other and object to an unknown expectation. The objective of a CMAB problem is to find a subset of all arms under specified constraints so

that the sum of the reward is maximized. In our scenario, each participant is an arm for a particular sensing task. Thus we have 2^n possible subsets and each subset is considered as a super arm. For each task, a super arm is played to perform this task. If an arm belongs to an selected super arm, its utility will be revealed after performing the task. Obviously, the utility of a super arm for a given arm is a function of the arm set of this super arm and the utility expectation of each arm.

Theorem 3. The expected utility of playing any super arm $P \subseteq \mathbb{P}$ is monotonically nondecreasing with respect to the expected utility of each participant for a give task $s_k(l(s_k), t(s_k))$, i.e., if for all $p_i \in P$, $U(p_i, l(s_k), t(s_k)) < U'(p_i, l(s_k), t(s_k))$, we have $\mathbf{U}(P, l(s_k), t(s_k)) < \mathbf{U}'(P, l(s_k), t(s_k))$

Proof. Assume there is an arbitrary selected participant set P for s_k, then we have $\mathbf{U}(P, l(s_k), t(s_k))$ is equals to $1 - \Pi_{i \in P}(1 - U(p_i, l(s_k), t(s_k)))$ while $\mathbf{U}'(P, l(s_k), t(s_k))$ is equals to $1 - \Pi_{i \in P}(1 - U'(p_i, l(s_k), t(s_k)))$. We have $1 - U(p_i, l(s_k), t(s_k)) > 1 - U'(p_i, l(s_k), t(s_k))$ and $\Pi_{i \in P}(1 - U(p_i, l(s_k), t(s_k)) > \Pi_{i \in P}(1 - U'(p_i, l(s_k), t(s_k)))$ for any $p_i \in P$ since $U(p_i, l(s_k), t(s_k)) < U'(p_i, l(s_k), t(s_k))$. Therefore the theorem holds. □

With the above in mind, the overall online self learning algorithm conceptually proceeds as follows. After some initialization processes, we could maintain an empirical mean of the capability $(\tilde{\boldsymbol{Pr}}_i, \tilde{\tau}_i)$ for each participant p_i. This empirical mean could be maintained by the methods (Equations (1) and (2)) discussed in Section 6.2.2. The real expectation of the two capability parameters are still unknown but estimated by $\bar{U}(p_i, l(s_k), t(s_k))$ from their empirical values $\tilde{U}(p_i, l(s_k), t(s_k))$ with an adjustment (Algorithm 1, Line 14-15) after each round of participant selection (done at Algorithm

1, Line 4). We estimate the empirical values $\tilde{U}(p_i, l(s_k), t(s_k))$ of each participant p_i for a given task $s_k(l(s_k), t(s_k))$ based on the output from the task performance (Algorithm 1, Line 7-14). Note depending on whether p_i visit the location at required time and whether the sensing data is uploaded to the platform, the capability $(\tilde{\boldsymbol{Pr}}_i, \tilde{\tau}_i)$ are updated. Since the empirical value is calculated by the number of the times that an arm is selected, the estimated expected capability depends on the number of historical tasks k and the number of the times $N_k(p_i, l(s_k), t(s_k))$ that an arm is selected. Here, we use $N_k(p_i, l(s_k), t(s_k))$ to record the number of time that p_i is selected to perform tasks at $l(s_k)$ and $t(s_k)$ until kth-round. Algorithm 7 shows the detailed steps.

Algorithm 8 and Algorithm 9 show two simple greedy methods to select a super arm in each round. In each time, they add the participant who contributes either the largest utility improvement (Algorithm 2) or the largest ratio between her utility improvement and cost (Algorithm 3), until the budget is not allowed, respectively. Note that our PSP problem is a submodular maximization problem with Knapsack constraints. Either Algorithm 2 and Algorithm 3 may not give good approximation ratio of the problem, but the best of these two algorithms can indeed [78]. Therefore, in Algorithm 1, Line 4, we pick the better solution out of these two greedy algorithm.

Traditional online learning algorithms for solving MAB problems has two phases, *exploration* and *exploitation*. In exploration phase, some arms are selected to evaluate their efficiency. In exploitation phase, the best performed arms are selected so that the total revenue is maximized. In both exploration and exploitation phases, the revenue is found after the arms are selected. Here, in our solution (Algorithm 1 and 2), these two phases are taken simultaneously, which means that each exploitation

Algorithm 8 Greedy Selection 1 for Task s_k

Input: Task $s_k(l(s_k, t(s_k))$, and utility $\bar{U}(p_i, l(s_k), t(s_k))$ and price r_i for each $p_i \in \mathbb{P}$
Output: P_k for s_k in this round

```
P_k = ∅ and θ_{i,k} = 0 for all i
while Σ_{i∈[1,n]} r_i θ_{i,k} ≤ B do
    for all p_i ∈ P and θ_{i,k} = 0 do
        Calculate the improvement of U(P_k, l(s_k), t(s_k)) (based on Equation (5)) by adding p_i into P_k
    end for
    Select the user p_i who leads to the largest utility improvement, set θ_{i,k} = 1 and add it into P_k
end while
return P_k
```

Algorithm 9 Greedy Selection 2 for Task s_k

Input: Task $s_k(l(s_k, t(s_k))$, and utility $\bar{U}(p_i, l(s_k), t(s_k))$ and price r_i for each $p_i \in \mathbb{P}$
Output: P_k for s_k in this round

```
P_k = ∅ and θ_{i,k} = 0 for all i
while Σ_{i∈[1,n]} r_i θ_{i,k} ≤ B do
    for all p_i ∈ P and θ_{i,k} = 0 do
        Calculate the improvement of U(P_k, l(s_k), t(s_k)) (based on Equation (5)) by adding p_i into P_k
    end for
    Select the user p_i who leads to the largest ratio between its utility improvement and cost r_i, set θ_{i,k} = 1 and add it into P_k
end while
return P_k
```

phase is also an exploration phase. We use the exploitation results to perform the exploration, i.e. update the capability of each selected participant in exploitation.

6.3.2 Approximation over Optimal Selection

The proposed algorithms are online process in which the estimated capability metrics of participants are updated by time. For each round of selection, Algorithm 2 determines the selected set of participants based on the latest estimation. Given a budget constraint B of selected participants for each task s_k, there should be a combinatorial number of possible selection to decide the optimal selection of participants.

As we showed in Theorem 1, this participant selection problem is still NP hard even if the true values of the capability metrics are known. Fortunately, we can prove that the proposed algorithm can achieve efficiently approximation ratio for the PSP.

First, we prove the utility is submodular.

Theorem 4. The utility $\mathbf{U}(\cdot)$ used in our greedy selection in Algorithm 2 or 3 is submodular.

Proof. Let let X and Y be two subset of participant selections, and $X \subseteq Y \subseteq \mathbb{P}$ and $e \in \mathbb{P} - Y$. For simplicity, we use $\mathbf{U}(\cdot)$ to represent $\mathbf{U}(\cdot, l(s_k), t(s_k))$ and $U(\cdot)$ to represent $U(\cdot, l(s_k), t(s_k))$.

As Equation (5), we have $\mathbf{U}(X) = 1 - \Pi_{p_i \in X}(1 - U(p_i)$ and $\mathbf{U}(Y) = 1 - \Pi_{p_i \in Y}(1 - U(p_i)) = 1 - \Pi_{p_i \in X}(1 - U(p_i)\Pi_{p_j \in Y - X}(1 - U(p_j))$. In addition we have:

$$\begin{aligned} \mathbf{U}(X \cup \{e\}) - \mathbf{U}(X) &= 1 - \Pi_{p_i \in X}(1 - U(p_i))(1 - U(e)) \\ -(1 - \Pi_{p_i \in X}(1 - U(p_i)) &= U(e)\Pi_{p_i \in X}(1 - U(p_i)) \end{aligned} \tag{15}$$

and

$$\begin{aligned} \mathbf{U}(Y \cup \{e\}) - \mathbf{U}(Y) &= U(e)\Pi_{p_i \in Y}(1 - U(p_i)) \\ &= U(e)\Pi_{p_i \in X}(1 - U(p_i)\Pi_{p_j \in Y - X}(1 - U(p_j)). \end{aligned} \tag{16}$$

Therefore,

$$\begin{aligned} &\mathbf{U}(X \cup \{e\}) - \mathbf{U}(X) - (\mathbf{U}(Y \cup \{e\}) - \mathbf{U}(Y)) \\ &= U(e)\Pi_{p_i \in X}(1 - U(p_i))(1 - \Pi_{p_j \in Y - X}(1 - U(p_j)). \end{aligned} \tag{17}$$

Since all the three factors in the last result are larger than or equal to 0. Therefore we have $\mathbf{U}(X \cup \{e\}) - \mathbf{U}(X) - (\mathbf{U}(Y \cup \{e\}) - \mathbf{U}(Y)) \geq 0$. Then $\mathbf{U}(\cdot)$ is submodular. □

Theorem 5. [78] For a non-negative, monotone submodular function f and a set of

element S in which each element has a cost $c(s)$, let P^* be a set that maximizes the value of f over all sets whose cost satisfies the budget constraint B. Let P_1 be a set obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the function value while B allows. Let P_2 be a set obtained by selecting elements one at a time, each time choosing an element that provides the largest ratio between the marginal increase in the function value and its cost while B allows. Then $\max(f(P_1), f(P_2)) \geq (1 - 1/e)/2 \cdot f(P^*)$; in other words, $\max(f(P_1), f(P_2))$ provides an $(1 - 1/e)/2$ approximation.

Based on Theorem 2, Theorem 3 and Theorem 4, we know that Algorithm 1 can achieve a $(1 - 1/e)/2$ approximation for PSP in each round. Next theorem also gives the time complexity of our algorithms.

Theorem 6. The total time complexity of proposed algorithms is $O(mn^2)$.

Proof. For each selection P_k for a given task s_k, Algorithm 2 or Algorithm 3 needs to calculate each participant's ratio of utility improvement and cost for current P_k. Thus, the total complexity of Algorithm 2 or Algorithm 3 is $O(n^2)$. Since there is m tasks in Algorithm 1, the total time complexity is $O(mn^2)$. □

6.3.3 Regret Analysis

With an α-approximation algorithm, it is no longer fair to compare the performance of a CMAB algorithm against the optimal reward as the regret of the algorithm. Instead, we can compare against the α-fraction of the optimal reward, because the reward is only α-approximate of the optimal value. In other words, we define α-*approximation regret* of a CMAB algorithm $\mathbb{A}$ after m rounds of play using an α-

approximation oracle is the difference between $m \cdot \alpha \cdot \mathbf{U}_{opt}$ and the expected cumulative utility obtained by algorithm $\mathbb{A}$. To perform the regret analysis of the proposed algorithm, we also need the following definitions.

Definition 7. For a given participant p_i, a task $s_k(l(s_k), t(s_k))$ and an optimal utility $\mathbf{U}_{opt}$, we define $P_{bad} = \{P' | \mathbf{U}(P', l(s_k), t(s_k)) < (1 - 1/e)/2\mathbf{U}_{opt}\}$ as the set of bad super arms. Then we define

$$\begin{aligned} \Delta^i_{min} = (1 - 1/e)/2\mathbf{U}_{opt} - max\{\mathbf{U}(P', l(s_k), t(s_k)) | P' \in P_{bad}, \\ p_i \in P'\} \\ \Delta^i_{max} = (1 - 1/e)/2\mathbf{U}_{opt} - min\{\mathbf{U}(P', l(s_k), t(s_k)) | P' \in P_{bad}, \\ p_i \in P'\} \end{aligned} \tag{18}$$

Then we define: $\Delta_{min} = max_{p_i \in P} \Delta^i_{max}$ and $\Delta_{max} = min_{p_i \in P} \Delta^i_{min}$

Using the conclusion (Theorem 1) in [74], since we have an $(1 - 1/e)/2$ approximation oracle, we have the following theorem on regret bound of our proposed algorithms.

Theorem 7. With an $(1 - 1/e)/2$-approximation greedy algorithm (best from Algorithm 2 and Algorithm 3), the $(1 - 1/e)/2$ approximation regret of the proposed algorithm (Algorithm 1) for m tasks is at most

$$\sum_{p_i \in P, \Delta^i_{min} \geq 0} \frac{12n^2 \ln m}{\Delta_{min}} + (\frac{\pi^2}{3} + 1)n\Delta_{max}. \tag{19}$$

Note that in our proposed algorithms, the total number of the times (N_i) that a participant p_i is selected is larger than the number of times ($N_k(p_i, l(s_k), t(s_k))$) that a participant p_i is selected for any given task s_k since we have $N_i = \sum_{s_k \in S} N_k(p_i, l(s_k), t(s_k))$. Thus the theorem still holds.

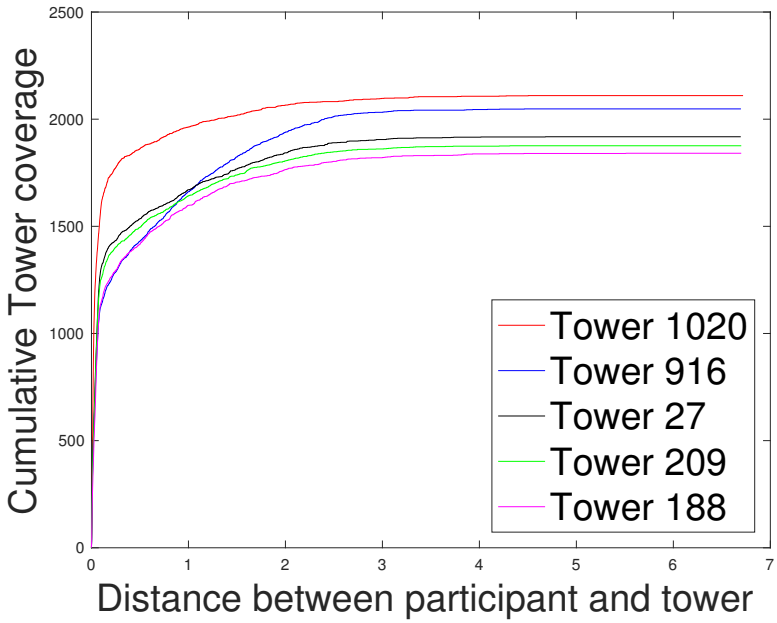


Figure 34: Cumulative tower coverage (visiting frequency) along the distance between a participant (her geometric center) and a tower for 5 towers in D4D dataset.

6.3.4 Handling Cold Start

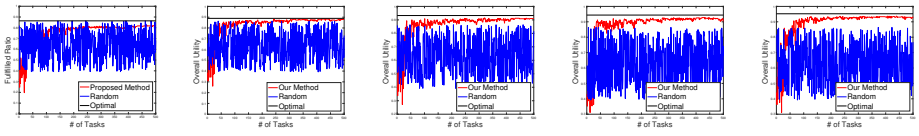
In our proposed algorithm, since initially the platform do not have any estimation on the capability of participant, we assign all of them equal to 1. Therefore, during the first round, the participant is randomly selected without any preference. However, such selection may be a very bad choice, and it may take longer time to converge to the good choice. This issue is named as a *cold start problem*, which exists in most of the learning methods. In this work, we also consider some possible solutions to handle the cold start problem.

Based on our observation, there is usually correlations among the adjacent locations in each participant's historical records. For example, if a participant appears in some locations of one area, then there is more chance that this participant visits the other locations in the same area. We analyze the historical records of some

participants in our dataset (more detail about the dataset is presented in next section) to verify this type of correlation. Figure 34 shows the cumulative visit times of 5 different locations (cellular towers) along with the distance between the participants and locations. For each participant, we utilize all her historical records to find the geometric center and then use the distance between this geometric center and the location as the distance between this participant and the location. Then we sort the distance between each participant and the location from short to long. Figure 34 shows that the visit frequency increases dramatically when the distance is short, while increases slowly when the distance is long. This indicates that the participants has a closer geometric center has more chance to visit a location than those who has further geometric centers. Therefore, we can use this rough estimation (the distance between her geometric center and the sensing target location) to scale the initial capability of a participant to certain sensing task. Another alternate way is using the home tower (most visited location) as the location of a participant. In our simulation section, we test both methods to handle the cold start problem.

6.4 Simulation Results

In this section, we conduct extensive simulations over a real-life mobile data to exam the effectiveness of our proposed algorithms under different scenarios. We mainly make the comparison among our proposed algorithm, a existing learning-based approach [73], and the optimal solution.



(a) $n = 50$ (b) $n = 100$ (c) $n = 150$ (d) $n = 200$ (e) $n = 250$

Figure 35: Results of our proposed algorithm compared with *Random* and optimal solution when $m = 500$ and $n = 50$ to 250.

6.4.1 Simulation Configuration

To simulate the large scale mobile crowd sensing (especially for mobile phone sensing), we use a real-life mobile dataset from the cellular operator Orange for the Data for Development (D4D) challenge [24]. We assume that the mobile users with the same user IDs are same users in all of these weekly call records. For the time delay distribution and cost value of each participant, we randomly generate them.

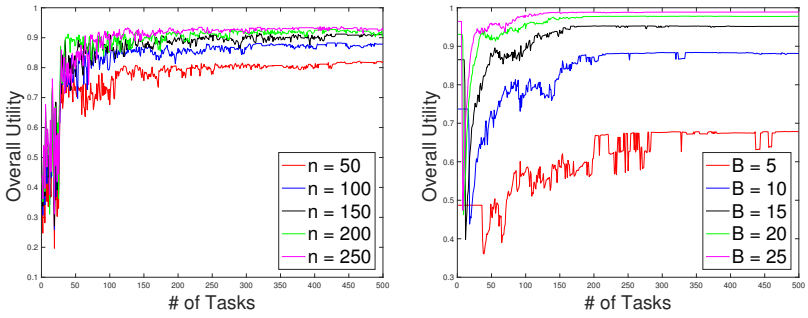
In all simulations, we randomly generate MCS tasks from all possible times and locations in each round. Then we apply our proposed algorithm on the tasks to select participants. The selected participants will perform the sensing with both spatial and temporal requirements and then have a fixed time limit to upload the sensing data. Once again, the real time delay is randomly drawn from the generated distribution. As discussed before, we assume that the whether a participant passes by the target location at target time is known. If a participant does not upload the sensing task in the time limit, it's delay in this round would be set to the time limit. However, if a participant does not pass by the target location at target time, only the sensing probability matrix will be updated for this participant in this round as shown in Algorithm 1. Note that here the optimization object is maximizing the expected utility of the selected participants, but not the real utility. Therefore, we measure

both expected utilities and real utilities on each round.

6.4.2 Performance Results

In the first set of simulations, we compare the performance of our proposed method among different number of candidate participants n. Here, we fix the budget B and generate the same type of sensing task for multiple rounds $m = 500$. We vary the number of candidate participants from 50 to 250. We also implement a random selection, *Random*, where a random set of participant is selected within budget limit in each round. Figure 35 shows the performance comparison. In each figure, we plot three curves which represent the optimal utility, the real overall utility of the selection by our method or *Random* in each task round. First, obviously, Random is not an efficient method, with very poor performances. Second, while the optimal utility keeps invariant, the real utility have a clear pattern in which it increases as the task round increases. As the task round increases, the number of times that each participant is selected increases. Thus, the estimated capability of each participant turns more accurate so that the selection has a trend to approach the optimal selection. This confirms the power of our proposed self-learning approach. Third, the real utility can not reach the optimal one, since the selection problem is NP-hard and our greedy algorithm is only approximation of the PSP. In other words, even if we have the absolute accurate estimation of the capability for each participant, it is still not guaranteed to achieve the optimal utility.

Figure 36(a) merges the results of our proposed methods in the first set of simulations. We can also find that as the number of candidate participants increases, the



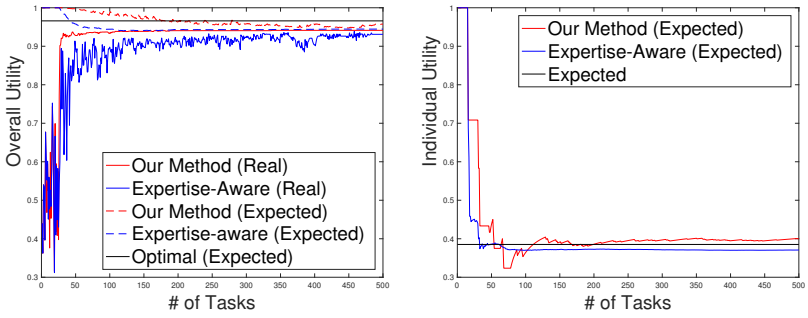
(a) various n (b) various B

Figure 36: Results of our proposed method for different n or B.

higher bound of overall utility increases as well as the optimal utility increases. This is reasonable because a larger participant pool means more possible better selection. Thus the real utility and optimal utility is monotonic along with the increasing of the number of candidate participants. In addition, Figure 36(a) also shows that the algorithm needs more time to converge as the number of candidate participants increases. The reason is that more candidate participants brings more possible selections thus it needs more rounds of selection for the algorithm to find better solutions.

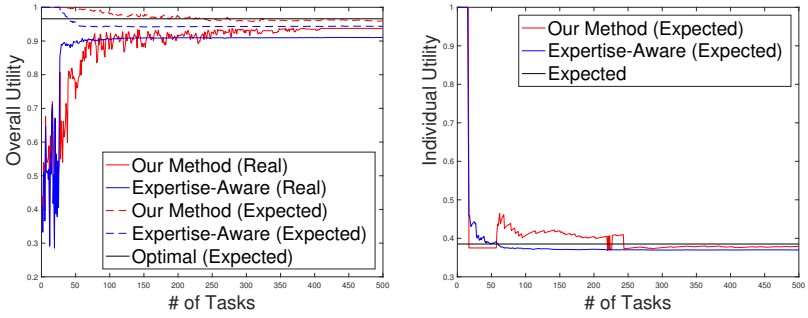
In the next set of simulations, we fix the number of candidate participant n and vary the budget B. Figure 36(b) shows the results. It is clear that the real utility of our algorithm increases as the budget B increases, since larger budget allows the algorithm have more chance to select efficient participants.

In the third set of simulations, we make comparisons against another expertise-aware solution [73], *Expertise-Aware* which is also based on self-learning techniques. It estimates and learns the expertise of each participant in performing sensing tasks. The main difference between it and our method is the way to learn capability (or expertise) of each participant. Our proposed method learns two type of expertise (sensing



(a) overall utility (b) individual utility

Figure 37: Results against *Expertise-Aware* when deadline = 24.

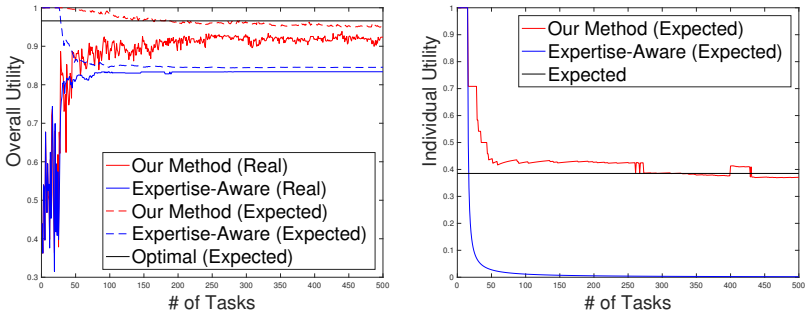


(a) overall utility (b) individual utility

Figure 38: Results against *Expertise-Aware* when deadline = 16

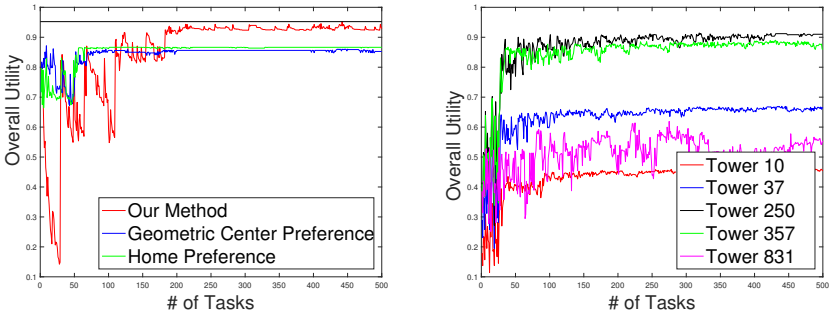
probability and time delay) separately, while *Expertise-Aware* learns the expertise as a single metric without considering the uploading deadline. Given different deadline requirements for the sensing tasks, the results vary for both methods. We run both algorithms with different deadlines, and results are reported in Figure 37 to Figure 39. In these figures, we plot both real utility and expected utility.

Figure 37(a) shows the results when the sensing deadline equals to 24. We can find that the two methods achieve similar performances of expected/real overall utility (of selected participants). The converge patterns are also similar. Figure 37(b) shows the individual expected utility (i.e., estimated capability/expertise) of one random selected participant. It converges to the true value with similar patterns in both of the methods. In summary, the two methods do not have significant difference when



(a) overall utility (b) individual utility

Figure 39: Results against *Expertise-Aware* when deadline = 8



(a) handling cold start (b) different types sensing tasks

Figure 40: (a) Results considering cold start; (b) Results for different sensing tasks (at different towers).

the deadline is long. Figure 38 shows the results when the sensing deadline equals to 16. Our proposed method begins to outperform *Expertise-Aware* by 4 percent. In addition, the estimated expected utility is more close to the true value against *Expertise-Aware* method. For results shown in Figure 39, the sensing deadline is set to 8. Our proposed method outperforms *Expertise-Aware* by 13 percent, while the estimated expected utility is much more accurate than *Expertise-Aware*. This is because the capability/expertise of our participants are modeled and updated separately, thus the updating of one of expertise will not be affected by the updating of others. It allows more detailed estimation and accurate learning results.

Next, we also test the two simple proposed ways to handle cold start. Figure 40(a)

shows the performances between our proposed method (without special handling) and our method with either location or home preference. Recall that we can utilize the geometric center of the participant's traces or the location with his highest visit frequency respectively to calculate the distance between one participant and the sensing task location. Then initially, we use such a distance to scale the default capability of each participant. From the simulation results we can see both proposed methods can make the convergence more smooth. However, the final results may not be better than the one without special handling.

Last, we also test different types of sensing tasks. For example, Figure 40(b) shows the performances of our proposed method on five different types of sensing tasks (here, they are different with five different target locations). It is obviously that the same set of participants have different capability for these sensing tasks. Some tasks can be performed better with this set of participants. This also confirms that the self-learning algorithm is necessary for various sensing tasks and various expertise from a diverse population.

6.5 Summary

In this work, we focus on a dynamic participant problem for heterogeneous mobile crowd sensing tasks, with a goal to maximize the sensing utility under budget constraints. Unlike other existing works, each participant's capability (with multi-expertise) is unknown to the selection mechanism. We propose an online algorithm to dynamically select a subset of participants to perform the tasks while updating the estimation of the capabilities of them. In our model, we consider both the sensing

probability from mobility pattern and the time delay from various uploading methods as participant?s capability. We provide detailed theoretical analysis of the proposed method with guaranteed approximation ratio and regret bound. Simulation results with a real-life dataset also confirm the efficiency of our proposed algorithms.

CHAPTER 7: CUMULATIVE PARTICIPANT SELECTION WITH SWITCH COSTS IN LARGE-SCALE MOBILE CROWD SENSING

7.1 Introduction

Smart mobile devices with various embedded sensors have been developing and deploying rapidly in recent years, which enables a new paradigm, *mobile crowding sensing* (MCS). In MCS, tremendous data can be collected by a large group of mobile device owners without additional infrastructure cost [7, 33]. There are advantages such as observing unpredictable events; dynamic temporal and spacial coverages; integrating human intelligence compared with traditional sensor networks. Therefore, MCS has been used in many applications such as human behavior mining [8, 9], environment monitoring [15, 16, 17], traffic planning [11, 12, 13, 14] and public safety monitoring [10]. Though MCS brings convenient, flexible and efficient solutions for heterogeneous sensing tasks, it also brings new challenges into the system design due to its large-scale tasks and participants. One of the most challenging problem in MCS is the *participant selection problem*.

It is challenging to select participants from a large candidates pool to perform heterogeneous sensing tasks. There are different constraints needed to be taken into consideration such as coverage [63, 34, 64], cost [18, 21, 65] and quality [66, 67]. It usually needs the participant information, such as the mobility patterns and sensing qualities, to select the appropriate participants for certain tasks more efficiently in

many real world applications. However, it is difficult to retrieve the participant information due to the lack of history records or privacy concerns [68, 69, 70, 82, 83]. Most of existing participant selection methods assume that the MCS platform knows the capability of each participant, either via historical records or self-reports from participants themselves. Some previous works simply assume that every one has the same capability to perform any task. However, in reality, the capability of a particular user for certain task depends on many factors, such as her moving pattern/behavior, device capability, sensor quality, or even uploading bandwidth. Therefore, how to learn the valued capability of the participants and utilize it to improve the performance of participant selection becomes a critical issue.

There are several studies begin to focus on such type of learning approaches, where the capability or quality is self learned from multiple trials over time [71, 72, 73]. However, [71] only considers learning sensing quality for homogeneous sensing tasks, where the sensing quality is modeled as random normalized variables. [72] focuses on the online labeling problem in which the true label is unknown. It only considers homogeneous sensing tasks with one-dimension metric. [73] consider heterogeneous tasks but learn the participant's expertise based on semantic analysis. Instead, the real heterogeneous tasks could have different temporal and spatial requirements, and the participant's capability could also include the data collection performance (such as time delay caused by different access methods or availabilities). Last, in certain applications, switching selected users also lead to additional cost (such as refreshing the security key or comprising the termination of a contract). None of the previous methods consider such *switch costs*. Therefore, in this work, we study the learning

method for a dynamic participant selection problem with additional consideration of these issues.

To address these issues, we formulate and investigate *cumulative participant selection problems* (with or without switch costs) in this work. In our MCS model, the selection mechanism selects the participants for heterogeneous sensing tasks on both spatial and temporal domain. The selected participant has unknown probability to perform the assigned sensing tasks. After performing sensing tasks, the participant may utilize different ways to upload collected sensing data, such as WiFi, Device-to-Device (D2D) or cellular networks. Different uploading methods may lead different delay time, which is still unknown by the selection mechanism. Having the historical records of participant performing tasks, we propose online learning algorithms to solve the participant problem by leveraging combinational multi-armed bandit (MAB) concepts. We show that the regret of proposed methods are bounded compared with optimal solution. Extensive simulations over real-life mobile datasets are conducted to evaluate the proposed algorithms in different MCS settings. Our results show the proposed methods can achieve great cumulative ultility. The main contributions of this work are summarized as below.

- We propose a new MCS system model to capture the *capability* of each participant in both sensing and data uploading stages for heterogeneous sensing tasks on both spatial and temporal domain.

- We formulate a *cumulative participant selection problem* (CPS) as a multi-armed bandit problem, and propose a corresponding online learning algorithm to guar-

antee bounded regret in theory.

- We introduce *switch costs* into MCS model, formulate a *cumulative participant selection problem with switch costs* (CPSS), and propose a corresponding online learning algorithm to theoretically guarantee a bounded regret.

- We implement our solutions over two real-world mobile datasets and conduct extensive simulations to evaluate the proposed methods against existing solutions.

The rest of this work is organized as follows. We first introduce our MCS system model and assumptions in Section 7.2. Then we present cumulative participant selection problem (without or with switch costs) and its corresponding proposed online participant selection algorithm in Section 7.3 and Section 7.4, respectively. Section 7.5 presents simulation results over two real-life mobile tracing datasets. Section **??** reviews the related works. Finally, Section 7.6 concludes this work.

7.2 System Model

7.2.1 MCS Model and Assumptions

As shown in Figure 41, there are three main components in the mobile crowd sensing system: a set of sensing tasks, a number of mobile participants, and a participant selection mechanism. The sensing tasks are generated by crowd sensing applications continuously and then sent to the participant selection mechanism. The mobile participants are mobile users, who are willing to participate in performing mobile sensing tasks. The participants must register with the MCS system, so that they could be

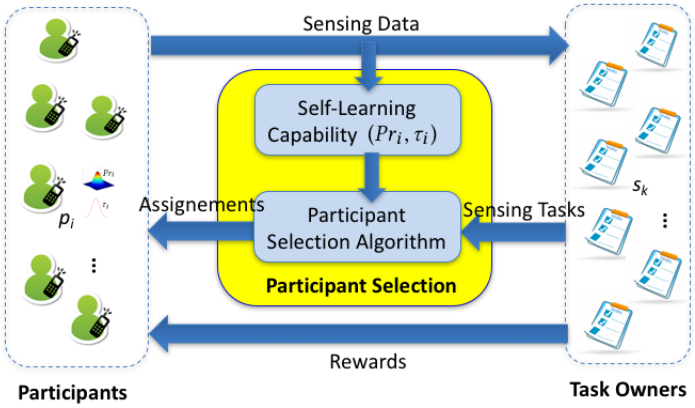


Figure 41: The framework of the proposed online learning for participant selection in MCS.

assigned with sensing tasks by the selection mechanism. The participant selection mechanism is a centralized mechanism in which the decision of which participant is selected to perform certain task is made. In this work, we assume that the task assignments can be sent to be selected participants via cellular networks instantly at any time. However, the collected sensing data may take more time to be sent back to the MCS applications due to its larger size and diverse collection paths (via WiFi or D2D relays).

7.2.1.1 Sensing Tasks

There is a sequence of M heterogeneous sensing tasks, denoted by $\mathbb{S} = \{s_1, s_2, ..., s_M\}$. The operation time of MCS is divided into time slots of equal length $t = \{1, 2, ...M\}$, for example, one day per time unit. Here, we assume that only one task is generated at the beginning of each time slot, but this can be relaxed to multi-task scenarios easily. Each sensing task s_k may require the sensing data in a location $l(s_k)$ at particular time $t(s_k)$ within the time slot. Given that we assume that each participant's

capability for different tasks are independent, we can simplify the model with heterogeneous tasks to one with only single type task. I.e., all tasks in $\mathbb{S}$ has the same target with the same spacial and temporal requirement. The participant selection problem with heterogeneous sensing tasks can be solved by dividing the original problem into multiple sub-problems with the same type sensing tasks.

Additionally, we also make the assumption that each sensing task has two different stages to be performed, *sensing stage* and *uploading stage*. In the *sensing stage*, the target sensing information of a task needs to be collected by selected participant(s) at the required location and time. In the *uploading stage*, the participant(s) who successfully collect the target sensing information need to upload the collected data to the MCS platform or applications. Note that previous works mostly only consider the sensing stage while ignore the uploading stage. However, in reality, if the sensing data is not successfully collected the sensing task should be considered a failure. This is one of the major contributions of this work.

7.2.1.2 Mobile Participants

There is a set of N participants, who are willing to participate into performing sensing tasks, denoted by $\mathbb{P} = \{p_1, p_2, ..., p_N\}$. These participants must register to the MCS platform so that they could be considered as candidates. Each participant p_i has her own *capability* to perform each specific type of sensing tasks. This *capability* can be treated as the success probability of the participant for such sensing tasks. Since we consider both sensing and uploading in our model, the capability can be represented by a tuple (Pr_i, τ_i) for a given type of tasks s_j, in which Pr_i represents

the expected mobility pattern of participant p_i (reach $l(s_j)$ at $t(s_j)$ during the *sensing stage*) and τ_i represents the expected time for p_i to upload the collected sensing data during the *uploading stage* after she succeeds collecting data.

This *capability* is critical and necessary knowledge for participant selection. However, the *capability* of each participant is unknown to the platform. We can only observe the values of those random variable outputs after each time the selected participants performing their sensing tasks. In this work, we assume that the participants are truthful and trusted (i.e., they will report if they fail to preform the sensing tasks).

7.2.1.3 Participant Selection Mechanism

In our proposed MCS system, MCS tasks arrive at the platform at each discrete time slot $t = 0, 1, ..., M$. In each time slot, the selection mechanism could communicate with the selected participants to assign MCS tasks to them. For each task s_k, the selection mechanism selects a subset P_k of $\mathbb{P}$ to perform it. Here we denote $\boldsymbol{\theta}_k = (\theta_{1,k}, \theta_{2,k}, ..., \theta_{N,k})$ as an indicator vector, in which $\theta_{i,k} \in \{0, 1\}$ indicates whether participant p_i is selected for task s_k. If so, $\theta_{i,k} = 1$; else $\theta_{i,k} = 0$. Furthermore, we define $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ...\boldsymbol{\theta}_M\}$ as all the selections made till task s_M. The goal of participant selection is to maximize the expected utility under budget constraint. The detailed definition of utility and problem formation are provided in Section 7.3 and 7.4.

7.2.1.4 Rewards

In this work, all the participants are associated with a global reward r, i.e., if a participant is selected for a task at certain time slot, it will be rewarded of r. In

addition, each switch of selected user may cost additional expense of the platform. For example, if participant p_i is selected to perform task s_j while not be selected for the next task s_{j+1}, then an additional compensation c has to be payed to p_i. This is because in real world applications, participants may have a contract with MCS platforms so that they could receive rewards for multiple tasks. If the platform decides to terminated this contract with a user, it has to pay an compensation to that user. Another possible switch cost is due to the additional initialization cost when the participant is selected, such as certain security initialization (e.g., key distribution). Note that previous works have never considered such switch costs, thus this is a unique contribution of this work.

7.2.2 Model Capability of Mobile Participants

As discussed above, there are two unknown attributes of capability of mobile participants in this work, one related to *mobility pattern during sensing stage* and the other related to *time delay during offloading stage*. We will discuss them respectively in the following subsections.

7.2.2.1 Sensing Probability

We use Pr_i to present the visiting pattern of each participant p_i over both spacial and temporal domains. We assume that participant p_i has her own probability Pr_i visiting the sensing target at $l(s_k)$ at certain time $t(s_k)$, we call this probability sensing probability. However, this probability is unknown to the platform. Denoted by $D_{i,\gamma}$ the random variable indicates whether participant p_i visits the sensing target when she is selected for the γth time. Then $D_{i,\gamma}$ is subject to a Bernoulli distribution

$B(1, Pr_i)$. We can still estimate this sensing probability Pr_i by leveraging the historical observations of $D_{i,\gamma}$. Here we assume the number of participant p_i being selected till time t is $N_{i,t}$. Then the estimated sensing probability $\tilde{Pr}_i$ can be obtained by the following:

$$\tilde{Pr}_i = \frac{\sum_{\gamma=1}^{N_{i,t}} D_{i,\gamma}}{N_{i,t}}. \tag{20}$$

7.2.2.2 Time delay

In this work, we assume that the selection mechanism can assign MCS tasks to participants at any time by sending a task request. After the selected participants perform the sensing tasks, they have to upload the collected data so that they can receive their rewards. There is a time delay before the participants complete uploading their sensed data. We define the time delay as the time from the ending of the sensing until the beginning of the data uploading as:

$$\tau = t_{Uploading} - t_{Sensing} \tag{21}$$

On one hand, some participants may not upload the sensed data through cellular networks, but wait for some low cost networks access (such as WiFi or D2D relays). On the other hand, it takes time to upload the data with a large size (note that the size of sensed data could be much larger than the size of sensing task). Different participants may have different network access patterns and bandwidths, thus each participant p_i has her own delay pattern on uploading MCS tasks. We assume that each participant p_i is associated with a set of random variables $T_{i,\gamma}$. Variable $T_{i,\gamma}$ indicates the random delay time of p_i in its γ-th selection. Each variable is independent

and subject to a known distribution with an unknown mean τ_i. To estimate the time delay of each participant p_i, we have:

$$\tilde{\tau}_i = \frac{\sum_{k=1}^{N_{i,t}} T_{i,k}}{N_{i,t}}. \tag{22}$$

This time delay represents the computing capability and encountering frequency to WiFi APs or D2D relays of each participant, which may vary among different tasks. The participants with shorter delay attribute has bigger probability to competing the uploading collected data faster than the ones with longer delay attribute.

7.2.2.3 Participant Utility

Denoted by $U_{i,\gamma}$, the utility of selecting a particular participant p_i to perform sensing task at her γth selection can be defined as follows:

$$U_{i,\gamma} = D_{i,\gamma}/T_{i,\gamma}. \tag{23}$$

Here, a higher sensing probability and a shorter time delay lead to higher utility. $U_{i,1}, U_{i,2}, ...$ is subject to an distribution $f(x, u_i)$ with unknown expectation u_i, in which $u_i = Pr_i/\tau_i$. Therefore, we have $\int_{-\infty}^{+\infty} xf(x, u_i)dx = u_i$.

Since the true values of Pr_i and τ_i are unknown and the estimations of them are updated along the time, we have an empirical $\tilde{Pr}_i$ and $\tilde{\tau}_i$ for each p_i at any time. Then we have the empirical expected utility $\tilde{u}_i$ at any time as:

$$\tilde{u}_i = \tilde{Pr}_i/\tilde{\tau}_i. \tag{24}$$

Theorem 8. The successive selection of each participant p_i products i.i.d. Participant

Utility.

Proof. Let $\mathbb{I} \in \mathbb{R}$ and $U_{i,s}$, $U_{i,t}$ be arbitrary pair of the random variables of p_i's utility output. Since they are subject to same distribution f, we have $P[u \leq U_{i,s}] = P[u \leq U_{i,t}]$ and $P[u \leq U_{i,t}] = P[u \leq U_{i,t} | u \leq U_{i,s}] \wedge P[u \leq U_{i,s}] = P[u \leq U_{i,s} | u \leq U_{i,t}]$, $\forall u \in \mathbb{I}$. Therefore they are identically distributed and independent (i.i.d). □

7.3 Cumulative Participant Selection

In this section, we propose the formulation of *Cumulative Participant Selection Problem* (CPS), in which we focus on the maximization of the cumulative utility for multiple tasks. Particularly, we aim to solve the problem that maximizing the expected sum of the utilities for a give task set while satisfying the constraint on the number of selected participants.

7.3.1 Problem Formulation

We firstly define the expectation of cumulative utility $J(t)$ till time t as follows:

$$E(J(t)) = \sum_{i=1}^{N} u_i E(N_{i,t}), \tag{25}$$

where $N_{i,t}$ is the number that participant p_i is selected till time t. Then the CPS problem we try to solve is defined as follows.

Definition 8. **Cumulative Participant Selection (CPS)**. Given the volunteering participants $\mathbb{P}$ and the crowd sensing task set $\mathbb{S}$, the CPS tries to make all selections

$\mathbf{\Theta}^*$ for all task $s \in \mathbb{S}$ with the objective:

$$\begin{aligned} &\max_{\mathbf{\Theta}^*} E(J(M)) \\ &s.t. \quad \sum_{i \in [1,N]} \theta_{i,j} = n \quad \text{for each} \quad j \in [1, M]. \end{aligned} \tag{26}$$

Note that for each time slot, only n participants are selected.

The CPS problem we define above belongs to the concept of *Multi-armed Bandit Problem* (MAB). In classical MAB problems, there are a number of arms for a player to play. The playing of each arm provides a random reward from a specific distribution. The player's goal is to maximize the rewords through a sequence of playing of the arms. In CPS, the participants can be regarded as different arms and the participant utilities are the rewards provided through the selections of participants.

Definition 9. **n-best participants and n-worst participants**. We first let σ be a permutation of $(1, ..., n, ...N)$ such that

$$u_{\sigma(1)} \geq u_{\sigma(2)} \geq ... \geq u_{\sigma(n)} \geq ... \geq u_{\sigma(N)} \tag{27}$$

If $u_{\sigma(n)} > u_{\sigma(n+1)}$, we call $\sigma(1)$,... $\sigma(n)$ the distinct n-best participants and $\sigma(n+1)$, ... $\sigma(N)$ the distinct n-worst participants.

Else if $u_{\sigma(n)} = u_{\sigma(n+1)}$, $0 \leq l < n$ and $n \leq k \leq N$ such that $u_{\sigma(1)} \geq ... \geq u_{\sigma(\sigma(l))} > u_{\sigma(l+1)} = ... = u_{\sigma(n)} = ... = u_{\sigma(k)} > u_{\sigma(k+1)} \geq ... \geq u_{\sigma(N)}$, we call $\sigma(1)$, ... $\sigma(l)$ the distinct n-best participants and $\sigma(k+1)$, ... $\sigma(N)$ the distinct n-worst participants.

Then we call participants with expectation utility equal to $u_\sigma(n)$ the n-border participants.

In most MAB models, regret is used to evaluate the performance of the given solution. The regret is defined as the difference between the given solution and the optimal solution. Here the optimal solution should be choosing the n-best participants for each sensing task. Then we have the regret for any given selection until t:

$$R_t = t \sum_{i=1}^{n} u_{\sigma(i)} - E(J(t)). \tag{28}$$

The first part of the regret represents the optimal solution, in which we always select the n-best participant for all tasks. The second part of the regret represents our problem objective. Therefore, our goal can be re-formalized to minimize the regret defined above at M.

7.3.2 Regret Analysis

Before we provide our solution to the CPS problem, we first have some analysis on the regret. Given the Kullback-Liebler number,

$$I(\theta, \lambda) = \int_{-\infty}^{+\infty} log[\frac{f(x,\theta)}{f(x,\lambda)}] f(x,\theta) d(x), \tag{29}$$

which is the measure of dissimilarity between two distributions. Based on the assumption made in Section 7.2, we have the following three conclusions.

$$0 < I(\theta, \lambda) < \infty \text{ if } \lambda > \theta, \tag{30}$$

$$I(\theta, \lambda) \text{ is continuous in } \lambda > \theta \text{ for fixed } \theta, \tag{31}$$

$$\text{and for all } u \text{ and } \delta > 0, \exists u \text{ s.t. } u < u^{'} < u + \delta. \tag{32}$$

Here we consider a uniformly good selection rule such that $R_t = o(t^{\alpha})$ for every

real $\alpha > 0$. According to Theorem 3.1 in [85] we have the following result.

Theorem 9. Let Θ be uniformly good rule. For each distinctly n-worst participant p_i and each $\epsilon > 0$, we have:

$$\liminf_{t\to\infty} \frac{E(N_{i,t})}{\log t} \geq \frac{1}{I(u_i, u_{\sigma(n)})}, \tag{33}$$

and consequently,

$$\liminf_{t\to\infty} \frac{R_t}{\log t} \geq \sum_{\text{i is n-worst}} \frac{u_{\sigma(n)} - u_i}{I(u_i, u_{\sigma(n)})}. \tag{34}$$

Theorem 9 proposes the estimation of the number of selection of an n-worst participant till time t and a lower bound of the total regret. Using the conclusion in Theorem 9, we have an *asymptotically efficient* selection defined as:

$$\limsup_{t\to\infty} \frac{R_t}{\log t} \leq \sum_{\text{i is n-worst}} \frac{u_{\sigma(n)} - u_i}{I(u_i, u_{\sigma(n)})}. \tag{35}$$

Such an asymptotically efficient selection is the solution we try to find for our proposed CPS problem.

7.3.3 Online Learning Algorithm

To solve the CPS, we need to determine when we should test some participants or select the participants with the best performance at any given time t. MAB algorithms usually consist two phases, *exploitation* and *exploration*. In the exploitation phase, arms are selected based on the current experiences. On the contrary, arms are selected to be tested in the exploitation phase. On one hand, we could only select the arms based on the formal performance of the arms since the arm output distribution is unknown. On the other hand, the arm output is probabilistic thus sometimes the

'good' arm may have 'bad' outputs. Therefore, we need to explore arms to avoid missing 'good' arms. In this work, we select the n participants who have the highest estimated utilities in exploitation phase. In exploration, we select some participant who do not have the highest estimated utilities but satisfy some constraints.

The main idea to solve this problem is to construct a family of statistics $g_{tN_{i,t}}$ for each participant p_i. When a participant's statistic is larger than the estimated utility of any n-best participants, we select this participant as an exploration for the next task.

According to [85, 84], we can construct the statistics as follows. Let $Y_1, Y_2, ..., Y_{N_{i,t}}$ be the utility output from a participant i. Let

$$W_{N_{i,t}}(u_i) = \int_{-\infty}^{0} \prod_{j=1}^{N_{i,t}} \frac{f(Y_j, u_i + t)}{f(Y_j, u_i)} h(t)dt, \tag{36}$$

where $h : (-\infty, 0) \to \mathbf{R}_+$ is a strictly positive continuous function with $\int_{-\infty}^{0} h(t)dt = 1$. For any $K > 0$ let

$$U(N_{i,t}, Y_1, Y_2, ..., Y_{N_{i,t}}, K) = \inf\{u | W_{N_{i,t}}(u) \geq K\}. \tag{37}$$

Here $W_{N_{i,t}}(u_i)$ serves as a natural statistic to test a group of samples generated from parameter u that whether $u < u_i$ or $u = u_i$ according to [84]. Here $W_{N_{i,t}}(u_i)$ increases as u_i increases. Therefore given a fixed K, we have the following conclusion. For any $u > U(N_{i,t}, Y_1, Y_2, ..., Y_{N_{i,t}}, K)$, the probability that the samples is generated from parameter $u_i < u$ is greater than the probability that the samples is generated form parameter u_i. For any $u < U(N_{i,t}, Y_1, Y_2, ..., Y_{N_{i,t}}, K)$. The probability that the samples is generated from parameter $u_i < u$ is smaller than the probability that

the samples is generated form parameter u_i. This property suggests that we can use $U(N_{i,t}, Y_1, Y_2, ..., Y_{N_{i,t}}, K)$ to determine when we should explore new participants. This also suggests us that the larger K is chosen, more accurate this estimation is.

Leveraging the above property, we let

$$g_{tN_{i,t}}(Y1, Y_2, ..., Y_{N_{i,t}}) = \mu[U(N_{i,t}, Y_1, Y_2, ..., Y_{N_{i,t}}, t(logt)^p)], \tag{38}$$

for some $p > 1$ as the statistics we construct. In addition we let

$$h_{N_{i,t}}(Y1, ..., Y_{N_{i,t}}) = \frac{Y_1 + Y_2 + ... + Y_{N_{i,t}}}{N_{i,t}}, \tag{39}$$

as the estimated mean utility of each participant p_i.

Let

$$U_t(i) = g_{tN_{i,t}}(Y_1, Y_2, ..., Y_{N_{i,t}}), \tag{40}$$

and

$$\mu_t(i) = h_{N(i,t)}(Y_1, Y_2, ..., Y_{N_{i,t}}), \tag{41}$$

be the statistic and estimate point for participant p_i at time t. Now we are ready to utilize the allocation rule proposed in [85] to solve the CPS in Algorithm 10 and Algorithm 11.

In Algorithm 10 we first select each participant n times thus we have an estimation of the utility of each participant. Then at any given time t, we determine whether we should select the same participants at $t-1$ or select a participant who is not among the n- best participants according to U_t and μ_t. If a participant's U_t is not less than the least best one of the n-best participants, we select this participant at

Algorithm 10 Online Learning Algorithm for CPS

Input: participant pool $\mathbb{P}$, task set $\mathbb{S}$.
Output: Θ^*

1: Initialization at $t = 0$, $\Theta^* = \emptyset$. Initialize $\tilde{\boldsymbol{P}}r$ and $\tilde{\boldsymbol{\tau}}$ to 0; denote the initialization task as s_0 and $N_{i,t} = 0$ and t = 0.
2: Choose $0 < \delta < 1/N^2$.
3: **for** each t **do**
4: **if** t = 0 **then**
5: Select each participant n times and update $\tilde{Pr}_i$ and $\tilde{\tau}_i$ for each participant p_i. Update t.
6: **else**
7: Calculate P_t^* using Algorithm 11.
8: Put P_t^* into Θ^* and update all $N_{i,t}$, $\tilde{P}r_i$ and $\tilde{\tau}_i$. $(i \in P_k^*)$.
9: $t = t + 1$.
10: **end if**
11: **end for**
12: Return Θ^*

time t instead.

Theorem 10. The selection described by Algorithm 10 and Algorithm 11 is *asymptotically efficient*.

Proof. We can prove the above theorem by proving the following setps and combining them together. Define $0 \leq l \leq n-1$ and $n \leq k \leq N$ by

$$u_1 \geq ... \geq u_l > u_{l+1} = ... = u_n = ... = u_k > u_{k+1} \geq ... \geq u_N, \tag{42}$$

and fixed $\epsilon > 0$, satisfying $\epsilon < u_l - u_n/2$ if $l > 0$ and $\epsilon < u_k - u_{k+1}/2$ if $k < N$.

- If $l > 0$ and $u_j \geq u_l$, then $E(t - N_{i,t}) = o(\log t)$.
- If $k < N$ and define the increasing sequence of integer-valued random variables $B_t = |\{N \leq a \leq t|$for some $j \geq n+1$ and j is one of the n-best participants at time $a + 1\}|$, then $E(B_t) = o(\log t)$.

Algorithm 11 Participant Selection at Time t

Input: participant pool $\mathbb{P}$, $\tilde{\boldsymbol{P}}r$ and $\tilde{\boldsymbol{\tau}}$.
Output: $P_t^* \in \mathbb{P}(|P_t^*| = n)$
1: Initialize $P_t^* = \emptyset$
2: **for all** $p_i \in \mathbb{P}$ **do**
3: Calculate $\mu_t(i)$ using current $\tilde{P}r_i$ and $\tilde{\tau}_i$.
4: **end for**
5: Sort the participants according to their estimate point $\mu_t(i)$ and let p_k be the n th participant.
6: Put first n participant into P_t^* .
7: Find $j \in \{1, 2, ..., N\}$ and $t + 1 \equiv j$ **mod** N
8: **if** $U_t(j) \geq \hat{\mu}_t(k)$ **then**
9: Replace p_k by p_j in P_t^*.
10: **end if**
11: Return P_t^*

- If $k < N$ and define the increasing sequence of integer-valued random variables $S_t(j) = |\{N \leq a \leq t|$All the n-bestparticipants at time $a + 1$ are among the participants$s$ with $u_s \geq u_k$ and for each n-best participant at time a+1, $|h_{N_{s,a}} - u_s| < \epsilon,$ but still participant p_j is selected at time $a + 1\}|$, then for each $\rho > 0$ we can choose $\epsilon > 0$ so small that $E(S_t(j)) \leq \frac{1+\rho+o(1)}{I(u_j,u_n)} \log t$.

The detailed proof of the above three steps can be found in Theorem 5.1 from [85]. □

Theorem 10 proves that by implementing the above selection method, we can achieve the regret bound given by Equation (16).

7.4 Cumulative Participant Selection with Switch Costs

In this section, we consider the cumulative selection problem with switch costs. In practical scenario, there maybe an additional cost if the selection mechanism switch the selected participants. This could be caused by the contract between the partici-

pants and the selection platform or the cost of each participant's own travel cost or initialization cost. In this scenario, we have to take this *switch cost* into account.

7.4.1 Problem Formulation

We first define a switch function of a selection $\theta_{i,k}$ for participant p_i and the kth task as:

$$d(\theta_{i,k}, \theta_{i,k+1}) = \begin{cases} 1 & \theta_{i,k} \neq \theta_{i,k+1}, \\ 0 & \theta_{i,k} = \theta_{i,k+1}. \end{cases} \tag{43}$$

Let

$$d(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k+1}) = \sum_{i=1}^{N} d(\theta_{i,k}, \theta_{i,k+1}) \tag{44}$$

be the total number of participant switches to task s_k and let

$$S_t = \sum_{k=1}^{t} d(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k+1}) \tag{45}$$

be the cumulative participant switches till time t. Then the objective of *Cumulative Selection Problem with Switch Cost* (CPSS) is defined as in the following definition.

Definition 10. **Cumulative Selection Problem with Switch Cost (CPSS)**. Given the volunteering participants $\mathbb{P}$ and the crowd sensing task set $\mathbb{S}$, for each task s_t ($s_t \in \mathbb{S}$), the participant selection problem tries to select a participant set P_t^* with the objective:

$$\begin{aligned} &\max_{\Theta^*} E(J(M) - cS_M) \\ s.t. \quad &\sum_{i \in [1,N]} \theta_{i,j} = n \qquad \text{for each} \quad j \in [1, M], \end{aligned} \tag{46}$$

in which c is the cost for each switch.

7.4.2 Regret Analysis

Similar to CPS, CPSS can be also reformed to a problem that minimizing the total regret of a given solution. The total regret of CPSS should be the sum of the utility regret and the switch regret. Thus we have:

$$R_t^{'} = R_t + cE(S_t). \tag{47}$$

Then our goal of CPSS is to minimize the $R_M^{'}$ above.

In Theorem 9 we can find that the number of times that we select any participant p_i who is not among the n-best participants is about $\frac{\log t}{I(u_i,u_{\sigma(n)})}$ until time t. Then we can estimate that the regret contribution for switching from this participant p_i is at most about $\frac{2\log t}{I(u_i,u_{\sigma(n)})}$, i.e. participant p_i is never successive selected more than once.

Through the analysis above we try to find an asymptotically efficient selection while satisfying the contribution for switching is much smaller than the upper bound $(o(\log t))$ above. A straight forward idea is to minimizing the number of switches when we select participants.

7.4.3 Online Learning Algorithm

Based on the idea in solving CPS, we try to determine whether a participant is well-estimated at any given time by using the number of the selections of this participant. Therefore, we could group the selection of any participant together to minimize the number of switches. We implement a block allocation method [86] to solve CPSS. The detailed algorithms to solve CPSS are given in Algorithm 12 and Algorithm 11.

The main idea of the proposed algorithm still includes two steps. In the first step,

Algorithm 12 Online Learning Algorithm for CPSS

Input: participant pool $\mathbb{P}$, task set $\mathbb{S}$, location set $\mathbb{L}$.
Output: Θ^*

Initialization at $t = 0$, $\Theta^* = \emptyset$. Initialize the $\tilde{\boldsymbol{P}}r$ and $\tilde{\boldsymbol{\tau}}$ to 0; denote the initialization task as s_0 and $N_{i,t} = 0$, let frame = 0, t = 0.
if frame = 0 **then**
  Select each participant N times.
else
  for each frame **do**
    Calculate the length of the frame and the blocks in this frame using Equations (48) and (49).
    if at the beginning of a block **then**
      Calculate P_t^* using using Algorithm 11.
      Put P_t^* into Θ^* and update all $N_{i,t}$, $\tilde{P}r_i$ and $\tilde{\tau}_i$. $(i \in P_k^*)$.
      $t = t + 1$.
    else
      Put P_{t-1}^* into Θ^*.
    end if
  end for
end if
Return Θ^*

we need to determine different time intervals, in which same participants are selected to perform the sensing tasks in these intervals. In the second steps, we need to determine which participants are selected during each interval.

Time is first divided into frames and each frame is further divided into blocks. Different frames have different lengths and while blocks in each frame have the same length. In each block, same participants are selected for performing sensing tasks. At the beginning of each block, the selection of participants is made. The length of the f frame is chosen as:

$$l_f = \begin{cases} N & f = 0 \\ \lfloor \frac{2^{f^2} - 2^{(f-1)^2}}{f} \rfloor Nf & f \geq 1, \end{cases} \tag{48}$$

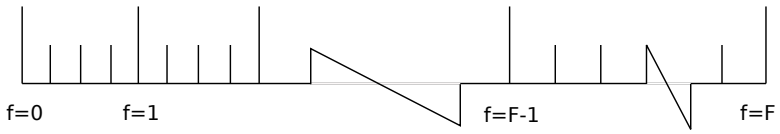


Figure 42: The frames allocation in Algorithm 12 when N = 4.

and the length of the block in frame f is:

$$b_f = \begin{cases} 1 & f = 0 \\ \\ f & f \geq 1. \end{cases} \tag{49}$$

Figure 42 shows an example of this frame allocation when $N = 4$. The way defining frames and blocks is mainly for proof of regret bound of the proposed algorithm.

Theorem 11. Assume that the participant have been re-indexed and $l \leq 0$ so that

$$u_1 \geq ... \geq u_l > u_{l+1} = ... = u_m > ... \geq u_p. \tag{50}$$

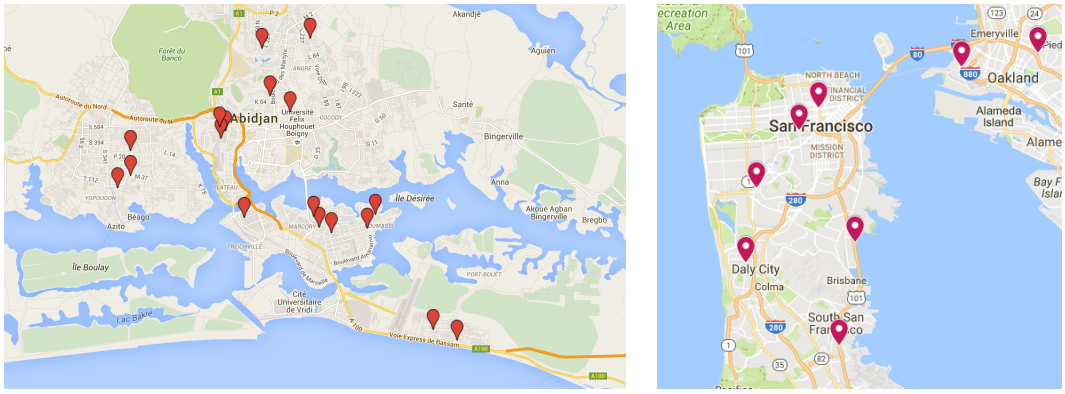
Under Θ^*, if there is a unique set of n-best arms amongst all of the N participants. Then we have:

$$\limsup_{t\to\infty} \frac{R_t'}{\log t} \leq \sum_{\text{i is n-worst}} \frac{u_{\sigma(n)} - u_i}{I(u_i, u_{\sigma(n)})}. \tag{51}$$

The proof of the above theorem can be found in [85] and [86]. This theorem indicates that the selection Θ^* is an asymptotically efficient selection. Therefore we can achieve the upper bound of the regret of the proposed algorithm as Equation (32) shows.

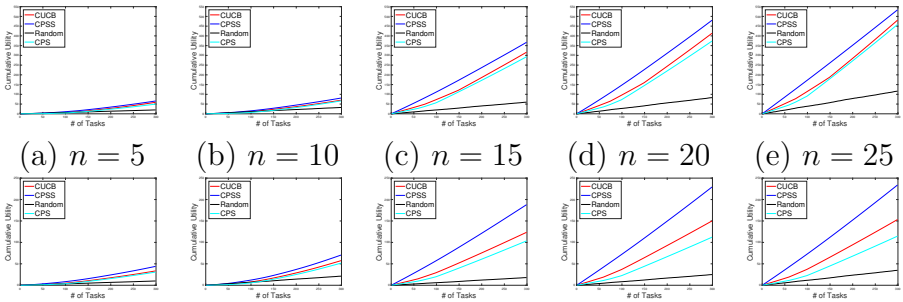
7.5 Evaluations

In this section, we conduct extensive simulations over real-life mobile data to exam the effectiveness of our proposed selection algorithms under different scenarios. Par-



(a) D4D (b) SFC

Figure 43: Locations of sensing targets: (a) cellular towers in Abidjan in D4D and (b) random GPS locations in SFC.



(a) $n = 5$ (b) $n = 10$ (c) $n = 15$ (d) $n = 20$ (e) $n = 25$

(f) $n = 5$ (g) $n = 10$ (h) $n = 15$ (i) $n = 20$ (j) $n = 25$

Figure 44: Results of different algorithms in D4D ((a)-(e)) and SFC ((f)-(j)) when $M = 300$, $N = 150$ and $n = 5$ to 25.

ticularly, we mainly implement the proposed algorithms and test them on two real-world datasets, *Data for development (D4D) dataset* [24] and *San Francisco Yellow Cab (SFC) dataset* [87].

7.5.1 Datasets and Simulation Configuration

7.5.1.1 D4D cellular dataset

D4D dataset [24] is utilized in the simulations. We use one day as the basic time unit to generate the sensing tasks. Different cellular towers are selected as the sensing target location to evaluate the proposed algorithms, as shown in Figure 43(a).

7.5.1.2 San Francisco Yellow Cab dataset

San Francisco Yellow Cab (SFC) dataset [87] collects the location information of yellow cabs in the city of San Francisco, in May 2008. The vehicles are all equipped with GPS tracking device. The GPS information of each vehicle is is transmitted to a central receiving station. From the records, we can extract the GPS position of each cab with a timestamp. There are 536 vehicles in total contained in this dataset. We randomly select GPS locations as the sensing targets as shown in Figure 43(b). We set a fixed sensing range of each location. If one taxi has one GPS record located in this sensing range, we consider this taxi visits this location once.

7.5.1.3 Ground Truth Generation

D4D dataset includes the cellular phone call records and FSC dataset includes the GPS records, by which we use to generate the ground truth of the visiting probability of each participant. Here, we assume that for each user we have multiple rounds of traces (e.g., K weeks), and there are K rounds in total . Let $c_k(p_i, l, t)$ indicate whether p_i made one or more phone call at l and t in D4D or visited l's range at least once at t in SFC (1 if it made, 0 otherwise). Then the ground truth of visiting probability (i.e. sensing probability Pr_i) can be generated as:

$$\tilde{Pr}(p_i, l, t) = \frac{\sum_{k=1}^{K} c_k(p_i, l, t)}{K} \tag{52}$$

In addition, the ground truth of the delay time τ_i is generated randomly in the simulations.

7.5.1.4 Other Simulation Settings

In all simulations, we randomly generate MCS tasks from the possible times and locations in each round. Then we apply our proposed algorithm on the tasks to select n participants per round. The selected participants will perform the sensing with both spatial and temporal requirements and then have a fixed time limit to upload the sensing data. As defined before, we assume that whether a participant passes by the target location at target time is known. If a participant does not upload the sensing task in the time limit, it's delay in this round would be set to the time limit. However, if a participant does not pass by the target location at target time. Only the sensing probability Pr_i will be updated for this participant in this round. Recall that the optimization object is to maximize the cumulative utility. In addition, we also implement the random selection and one existing method for MAB (CUCB [88]) to make comparison with our proposed algorithms.

7.5.2 Performance of Different Selection Algorithms

In the first set of simulations, we make the comparisons among proposed algorithms and other two base line algorithms. In total, four participant selection algorithms are implemented and compared.

- **CUCB** is an index-based algorithm which tries to solve the combinatorial multi-armed bandit (CMAB) problems proposed in [88]. However, it does not consider the switch cost in its design.

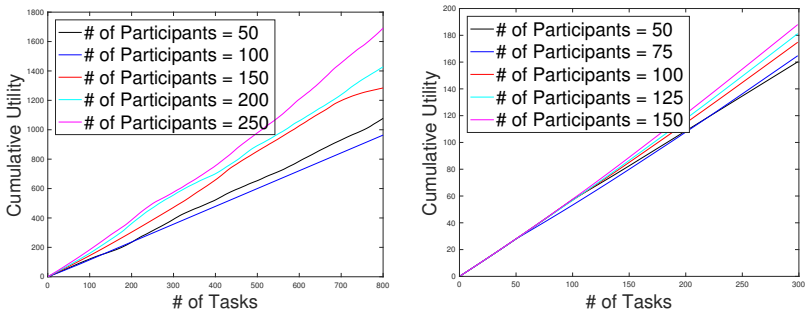
- **Random** simply selects n participants randomly from the candidate pool for each sensing task.
- **CPS** is the selection method proposed in Algorithm 10 and Algorithm 11, which does not consider the switch cost.
- **CPSS** is the selection method proposed in Algorithm 12 and Algorithm 11, which considerthe switch cost.

We set the budget of selection as n from 5 to 25, and perform all algorithms with $N = 150$ participants and $M = 300$ tasks. The results (the cumulative utility) are reported in Figure 44 for both D4D and SFC datasets. Similar trends can be found in both set of results. We can find that the proposed algorithm CPSS outperforms CUCB and Random. Specifically, CPSS outperforms CUCB about 15%. The CUCB can not achieve best utility per task at the beginning since it needs to switch the selected participants very frequently. Random also performs bad since every random selection has a large chance to switch many selected participants. CPS performs similar to CUCB and better than Random. In addition, as the number of selected participants (i.e. n) increases, the utility increases. This is reasonable since more selected participants intuitively lead to more utilities.

7.5.3 Performance of Proposed Algorithm over Different Settings

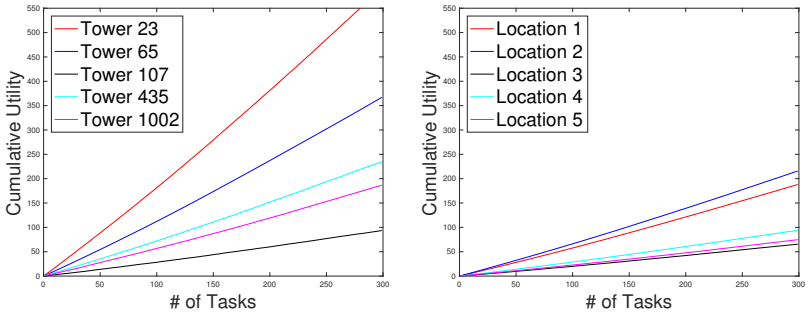
In the second set of simulations, we evaluate our proposed algorithm CPSS in different simulation settings.

First, we test with different size of candidate pool as shown in Figure 45. We



(a) D4D (b) SFC

Figure 45: Cumulative utility among different candidate pools.



(a) D4D (b) SFC

Figure 46: Cumulative utility among different locations.

vary the size of the candidate pool N while fix the number of selected participants $n = 15$. Here, the value of N varies from 50 to 250 for D4D and from 50 to 150 for SFC. The simulation results show that more candidates achieve better performance. This is because larger candidate pool contains more participants in which there are participants more competitive.

To demonstrate the differences of capability among different sensing tasks (i.e., different locations), we also plot results for 5 different locations as shown in Figure 46. Here we fix $N = 150$ and $n = 15$. The simulation results shows different participant patterns at different locations. which also lead to different utilities. However, the increasing trends are still similar, which shows that the algorithm works on different type of sensing tasks.

7.6 Summary

In this work, we focus on a dynamic participant problem for heterogeneous mobile crowd sensing tasks, with a goal to maximize the sensing utility under budget constraints. Unlike other existing works, each participant's capability is unknown to the selection mechanism. We propose an online algorithm to dynamically select a subset of participants to perform the tasks while updating the estimation of the capabilities of them. In our model, we not only consider both the sensing probability from mobility pattern and the time delay from various uploading methods as participant's capability, but also can take the cost of switching participants into account. We provide both theoretical regret analysis and extensive simulations conducted with real-life datasets. Simulation results confirm the efficiency of our proposed algorithms. We leave further improvements on our online learning algorithms (such as design a cold start strategy for our proposed algorithm to accelerate the convergence) as one of our future works.

CHAPTER 8: CONCLUSION

As we address, one of the key challenges in MCS participant selection is how to effectively select appropriate participants from a huge user pool to perform various sensing tasks while satisfying certain constraints. Firstly, there has to be specified MCS architectures in which the working processes can be organized. Secondly, the crowd sensing tasks are usually heterogeneous that they may have different requirement in both spatial and temporal domains. Meanwhile, MCS systems often run in dynamic environments in which the patterns of both participants and tasks vary with time and locations. In addition, we also need to consider the data transmission mechanisms in MCS systems.

In this dissertation, we present different mechanisms to enhance the performance of participant selection and task assignment in MCS. We try to solve the participant selection problem from the different aspects mentioned above respectively. For each scenario, we proposed detailed system model and problem definition. Then we present our solution and essential analysis. Extensive simulations over a real-life mobile dataset confirm the efficiency of the proposed algorithms in these works.

In our future works, we plan to investigate hybrid data collection schemes which combine D2D and direct communications to deliver the collected sensing data. This is because different data accesses may have different time or reward cost. The participants may also have their own preferences for data delivery so that they can optimize

their utility. In addition, we consider adding energy consumption model in current mobile crowd sensing model so that we can design energy efficient solutions. Finally, we plan to utilize multiple real world datasets since each dataset has its own features thus different datasets may bring different inspirations in solving the participant selection problem.

REFERENCES

[1] F.-Y. Wang, "The emergence of intelligent enterprises: From CPS to CPSS," *IEEE Intelligent Systems*, 25(4):85-88, 2010.

[2] A. Sheth, P. Anantharam, et al., "Physical-cyber-social computing: An early 21st century approach," *IEEE Intell. Syst.*, 28(1):78-82, 2013.

[3] J. Zeng, et al., "A system-level modeling and design for cyber-physical-social systems," *ACM Trans. Embedded Comput. Syst.*, 15(2):35, 2016.

[4] Y. Wang, et al., "Mobile big data meets cyber physical system: Mobile crowd-sensing based cyber-physical system for smart urban traffic control," in *2015 Workshop for Big Data Analytics in CPS*, 2015.

[5] C. Huang, et al., "Towards reliable social sensing in cyber-physical-social systems," in *W. on Par. & Dist. Proc. for Comp Social Sys.*, 2016.

[6] C. Bo, et al., "Detecting driver's smartphone usage via non-intrusively sensing driving dynamics," *IEEE Internet of Things Journal*, to appear.

[7] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32-39, 2011.

[8] N. Lathia, V. Pejovic, K. Rachuri, C. Mascolo, M. Musolesi, and P. Rentfrow, "Smartphones for large-scale behavior change interventions," *IEEE Pervasive Computing*, vol. 12, no. 3, pp. 66–73, July 2013.

[9] A. Noulas, S. Scellato, R. Lambiotte, M. Pontil, and C. Mascolo, "A tale of many cities: universal patterns in human urban mobility," *PLOS ONE*, vol. 7, no. 5, p. e37027, 2012.

[10] 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*, vol. 8, no. 8, p. e1001083, 08 2011.

[11] Y. Wang, X. Liu, H. Wei, G. Forman, and Y. Zhu, "Crowdatlas: Self-updating maps for cloud and personal use," in *Proceeding of the 11th Annual ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013.

[12] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing," in *Proceedings of the 10th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2012.

[13] V. Coric and M. Gruteser, "Crowdsensing maps of on-street parking spaces," in *Proceedings of the 2013 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2013.

[14] S. Nawaz, C. Efstratiou, and C. Mascolo, "Parksense: A smartphone based sensing system for on-street parking," in *Proceedings of the 19th ACM International Conference on Mobile Computing and Networking (MOBICOM 2013)*, 2013.

[15] N. Maisonneuve, M. Stevens, and B. Ochab, "Participatory noise pollution monitoring using mobile phones," *Info. Pol.*, vol. 15, no. 1,2, pp. 51–71, Apr. 2010.

[16] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: An end-to-end participatory urban noise mapping system," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2010.

[17] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "PEIR, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proceedings of the 7th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2009.

[18] H. Xiong, D. Zhang, L. Wang, J. Gibson, and J. Zhu, "EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants," *ACM Transactions on Intelligent Systems and Technology (TIST)*, to appear, 2015.

[19] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *IEEE International Conference on Pervasive Computing and Communications(Percom'15)*, 2015.

[20] D. Zhang, H. Xiong, L. Wang, and G. Chen, "Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *The 2014 ACM Conference on Ubiquitous Computing*, UbiComp '14, Seattle, WA, USA, 2014, pp. 703–714.

[21] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi, "EMC^3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint," *IEEE Transactions on Mobile Computing*, to appear, 2015.

[22] L. Pournajaf, L. Xiong, and V. S. Sunderam, "Dynamic data driven crowd sensing task assignment," in *Proceedings of the International Conference on Computational Science, ICCS 2014, Cairns, Queensland, Australia, 10-12 June, 2014*, 2014, pp. 1314–1323.

[23] L. Pournajaf, L. Xiong, V. S. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, Australia, July 14-18, 2014 - Volume 1*, 2014, pp. 73–82.

[24] V. D. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki, "Data for development: The d4d challenge on mobile phone data," in *arXiv.1210.0137v2*, 2013.

[25] The Data for Development (D4D) Challenge, http://www.d4d.orange.com.

[26] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, "Understanding the coverage and scalability of place-centric crowdsensing," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2013, pp. 3–12.

[27] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Proceedings of the 8th International Conference on Pervasive Computing (Pervasive)*, 2010, pp. 138–155.

[28] G. S. Tuncay, G. Benincasa, and A. Helmy, "Participant recruitment and data collection framework for opportunistic sensing: A comparative analysis," in *Proceedings of the 8th ACM MobiCom Workshop on Challenged Networks (CHANTS)*, 2013, pp. 25–30.

[29] D. Zhao, H. Ma, and L. Liu, "Energy-efficient opportunistic coverage for people-centric urban sensing," *Wireless Networks*, vol. 20, no. 6, pp. 1461–1476, 2014.

[30] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proceedings of the 18th Annual ACM International Conference on Mobile Computing and Networking (Mobicom)*, 2012.

[31] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos, "TRAC: truthful auction for location-aware collaborative sensing in mobile crowdsourcing," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, 2014, pp. 1231–1239.

[32] D. Zhao, X. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, 2014, pp. 1213–1221.

[33] 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.

[34] H. Li, T. Li, Y. Wang, "Dynamic participant recruitment of mobile crowd sensing for heterogeneous sensing tasks," in *Proc. of IEEE MASS*, 2015.

[35] S. Podlipnig and L. Boszormenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys*, 35(4):374–398, 2003.

[36] 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.

[37] D. Elsharief, H. Ibrahim, et al., "A survey of methods for maintaining mobile cache consistency," in *Proc. of ACM MoMM*, 2009.

[38] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015-2020 (February 3, 2016). http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-work-c11-520862.html.

[39] 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.

[40] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li, "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing," in *Proc. of ACM MobiCom*, 2014.

[41] C. Bo, T. Jung, X. Mao, X.-Y. Li, and Y. Wang, "SmartLoc: Sensing landmarks silently for smartphone based metropolitan localization," *EURASIP Journal on Wireless Communications and Networking*, 2016:e111, 2016.

[42] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting mobile 3G using WiFi. In *ACM MobiSys 2010*, 2010.

[43] S. Dimatteo, P. Hui, B. Han, and V.O.K. Li, "Cellular traffic offloading through WiFi networks," in *Proc. of IEEE MASS*, 2011.

[44] V. Chandrasekhar, J.G. Andrews, and A. Gatherer, "Femtocell networks: A survey," *IEEE Communications Magazine*, 46(9):59-67, 2008.

[45] B. Han, P. Hui, V.S.A. Kumar, M.V. Marathe, J. Shao, and A. Srinivasan. Mobile data offloading through opportunistic communications and social participation. *IEEE Transactions on Mobile Computing*, 11(5):821-834, 2012.

[46] Y. Li, M. Qian, D. Jin, P. Hui, Z. Wang and S. Chen, "Multiple mobile data offloading through disruption tolerant networks," *IEEE Trans. on Mobile Computing*, 13(7):1579–1596, 2014.

[47] Y. Zhu, C. Zhang, and Y. Wang, "Mobile data delivery through opportunistic communications among cellular users: A case study for the D4D challenge," in *Proc. of NetMob*, 2013.

[48] H. Li, T. Li, F. Li, W. Wang, and Y. Wang., "Enhancing participant selection through caching in mobile crowd sensing," in *Proc. of ACM/IEEE IWQoS*, 2016.

[49] L. Wang, D. Zhang, and H. Xiong, "Effsense: energy-efficient and cost-effective data uploading in mobile crowdsensing," in *Proc. of ACM UbiComp*, 2013.

[50] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proc. of IEEE INFOCOM*, 2015.

[51] H. Li, T. Li, X. Shi and Y. Wang, "Data collection through device-to-device communications for mobile big data sensing," in *Proceedings of 1st Workshop of Mission-Critical Big Data Analytics* (MCBDA 2016), 2016.

[52] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke Univ., Tech. Rep., 2000.

[53] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Tech. Rep. GIT-CC-04-07, 2004.

[54] M. Huang, S. Chen, Y. Zhu, and Y. Wang, "Topology control for time-evolving and predictable delay-tolerant networks," *IEEE Transactions on Computers*, 62(11):2308–2321, 2013.

[55] F. Li, S. Chen, M. Huang, Z. Yin, C. Zhang, and Y. Wang, "Reliable topology design in time-evolving delay-tolerant networks with unreliable links," *IEEE Trans. on Mobile Computing*, 14(6):1301–1314, 2015.

[56] A. Agrawal and R. E. Barlow, "A survey of network reliability and domination theory," *Operations Research*, 32:478–492, 1984.

[57] F. Li, C. Tian, T. Li, and Y. Wang, "Energy efficient social routing framework for mobile social sensing networks," *Tsinghua Science and Technology*, 21(4): 363–373, 2016.

[58] Y. Liu, F. Li, and Y. Wang, "Incentives for delay-constrained data query and feedback in mobile opportunistic crowdsensing," *Sensors*, 16(7), 2016.

[59] Y. Zhu, C. Zhang, F. Li, and Y. Wang, "Geo-Social: Routing with location and social metrics in mobile opportunistic networks," in *IEEE ICC*, 2015.

[60] Y. Liu, A.M.A.E. Bashar, F. Li, Y. Wang, and K. Liu, "Multi-copy data dissemination with probabilistic delay constraint in mobile opportunistic device-to-device networks," in *Proc. of 17th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (WOWMOM 2016), 2016.

[61] Y. Li, H. Wu, Y. Xia, Y. Wang, F. Li, and P. Yang, "Optimal online data dissemination for resource constrained mobile opportunistic networks," *IEEE Transactions on Vehicular Technology*, to appear.

[62] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A survey of social-based routing in delay tolerant networks: Positive and negative social effects," *IEEE Communication Survey and Tutorials*, 15(1):387–401, 2013.

[63] 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.

[64] H. Li, T. Li, et al., "Enhancing participant selection through caching in mobile crowd sensing," in *Proc. of IEEE/ACM IWQoS*, 2016.

[65] D. Zhao, H. Ma, and L. Liu, "Energy-efficient opportunistic coverage for people-centric urban sensing," *Wireless Net.*, 20(6):1461-1476, 2014.

[66] B. Guo, H. Chen, Q. Han, Z. Yu, D. Zhang, and Y. Wang, "Worker-contributed data utility measurement for visual crowdsensing systems," *IEEE Trans. on Mobile Computing*, 16(8): 2379-2391, 2017.

[67] M. Xiao, et al., "Secret-sharing-based secure user recruitment protocol for mobile crowdsensing," in *Proc. of IEEE INFOCOM*, 2017.

[68] K. L. Huang, et al., "Preserving privacy in participatory sensing systems," *Comput. Commun.*, 33(11):1266-1280, 2010.

[69] L. Pournajaf, L. Xiong, et al. "Spatial task assignment for crowd sensing with cloaked locations," in *Proc. of IEEE MDM*, 2014.

[70] H. Jin, L. Su, et al., "Enabling privacy-preserving incentives for mobile crowd sensing systems," in *Proc. of IEEE ICDCS*, 2016.

[71] H., Kai, C. Zhang, and J. Luo. "BLISS: Budget limited robust crowdsensing through online learning." in *Proc. of IEEE SECON*, 2014.

[72] Y. Liu and M. Liu. "An online learning approach to improving the quality of crowd-sourcing." IEEE/ACM Trans. on Networking (2017).

[73] X. Zhang, Y. Wu, et al.. "Expertise-aware truth analysis and task allocation in mobile crowdsourcing." in *Proc. IEEE ICDCS*, 2017.

[74] Chen, Wei, et al. "Combinatorial multi-armed bandit: General framework, results and applications." In *Proc. of ICML*, 2013.

[75] G. Gao, M. Xiao, et al., "Deadline-sensitive mobile data offloading via opportunistic communications," in *Proc. of the IEEE SECON*, 2016.

[76] M. Karaliopoulos, et al., "User recruitment for mobile crowdsensing over opportunistic networks," in *IEEE INFOCOM*, 2015.

[77] Y. Wang, H. Li, and T. Li, "Participant selection for data collection through device-to-device communications in mobile sensing," *Personal and Ubiquitous Computing*, 21(1):31–41, 2017.

[78] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, 3(19): 8, 2012.

[79] D. Zhao, X.-Y. Li, and H. Ma, "Budget-feasible online incentive mechanisms for crowdsourcing tasks truthfully," *IEEE/ACM Trans. on Networking*, 24(2):647-661, 2016.

[80] Z. Feng, Y. Zhu, et al., "TRAC: truthful auction for location-aware collaborative sensing in mobile crowdsourcing," in *Proc. of IEEE INFOCOM*, 2014

[81] H. Jin, L. Su, and K. Nahrstedt, "Centurion: Incentivizing multi-requester mobile crowd sensing," in *Proc. of IEEE INFOCOM*, 2017.

[82] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Inception: Incentivizing privacy-preserving data aggregation for mobile crowd sensing systems," in *Proc. of ACM MobiHoc*, 2016.

[83] T. Li, T. Jung, et al., "Scalable privacy-preserving participant selection in mobile crowd sensing," in *Proc. of IEEE PerCom*, 2017.

[84] M. Pollak and D. Siegmund. "Approximations to the expected sample size of certain sequential tests." The Annals of Statistics (1975): 1267-1282.

[85] Venkatachalam Anantharam, Pravin Varaiya, and Jean Walrand. "Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part I: IID rewards." IEEE Transactions on Automatic Control 32, no. 11 (1987): 968-976.

[86] R. Agrawal, M. Hegde, and D. Teneketzis. "Multi-armed bandit problems with multiple plays and switching cost." Stochastics: An International Journal of Probability and Stochastic Processes (1990).

[87] Michal Piorkowski and Natasa Sarafijanovoc-Djukic and Matthias Grossglauser, "A Parsimonious Model of Mobile Partitioned Networks with Clustering," in *The First International Conference on COMmunication Systems and NETworkS (COMSNETS)*, 2009.

[88] Wei Chen, Yajun Wang, and Yang Yuan. "Combinatorial multi-armed bandit: General framework, results and applications." In *Proceedings of the 30th International Conference on Machine Learning*, pp. 151-159. 2013.