Urban Sensing Based on Human Mobility

Shenggong Ji[1,*], **Yu Zheng**[2,1,3,†], **Tianrui Li**[1]
[1]Southwest Jiaotong University, Chengdu, Sichuan, China; [2]Microsoft Research, Beijing, China
[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
shenggongji@163.com, yuzheng@microsoft.com, trli@swjtu.edu.cn

ABSTRACT
Urban sensing is a foundation of urban computing, collecting data in cities through ubiquitous computing techniques, e.g. using humans as sensors. In this paper, we propose a crowd-based urban sensing framework that maximizes the coverage of collected data in a spatio-temporal space, based on human mobility of participants recruited by a given budget. This framework provides participants with unobstructed tasks that do not break their original commuting plans, while ensuring a sensing program balanced coverage of data that better supports upper-level applications. The framework consists of three components: 1) an objective function to measure data coverage based on the entropy of data with different spatio-temporal granularities; 2) a graph-based task design algorithm to compute a near-optimal task for each participant, using a dynamic programming strategy; 3) a participant recruitment mechanism to find a portion of participants from candidates for a given budget. We evaluate our framework based on a field study and simulations, finding its advantages beyond baselines.

ACM Classification Keywords
H.4.m Information Systems Applications: Miscellaneous

Author Keywords
Urban Computing; Urban Sensing; Crowdsensing

INTRODUCTION
Urban sensing is the foundation of urban computing [24], collecting data generated in urban spaces using different kinds of sensors. One type of urban sensing leverages people as sensors (i.e., crowd sensing [6]) to probe the physical world. In such crowd-based urban sensing applications, we hope the collected data can have a good coverage in both spatial and temporal spaces (called balanced data) so as to better support upper-level functions, such as monitoring [21], inferences [25] and predictions [27]. For example, to diagnose urban noises, we hope people participating in a sensing program can collect sound levels throughout an entire city and across different time intervals. Likewise, to better monitor traffic conditions of a city, we expect people traveling on different road segments can contribute their GPS trajectories at different time intervals. As human mobility is highly skewed [3] and uncertain [1, 7] in a city, however, it is difficult to design urban sensing methods that can result in data with a balanced coverage.

In this paper, given a budget, a geographical region and a time span, we design an urban sensing framework, which recruits participants and assigns tasks to them based on their mobility so as to maximize the coverage of collected data in the given spatio-temporal space. As illustrated in the left part of Figure 1, the owner of an urban sensing program can publish a geographical space (denoted by the broken red box) and a time span (e.g. 7-10am) where they are going to collect data, the type of data, as well as the minimal reward for each hour a participant can obtain. If interested in it, participants (e.g. u_1, u_2, u_3) submit the information about their mobility, consisting of an origin, a destination, a departure time, and an arrival time, before the sensing program starts. Our system estimates the slack time of each participant based on their mobility (i.e., the time span between a participant's arrival and departure times minus the necessary time needed for traveling between an origin and a destination). The system will then select proper participants (e.g. u_1 and u_2), based on their slack times and a budget specified by the program owner, to maximize the collective coverage of the data those participants will potentially collect in the spatio-temporal space. As shown in the right part of Figure 1, each participant is assigned a task that is comprised of a reward (r) and a sequence of collecting locations with corresponding time intervals. In case the data is not completely collected as planned (e.g. experiencing car accidents or traffic congestion), the system will conduct a second round of participant recruitment based on collected data (denoted by the tensor) and other participants' mobility, until the budget runs out or the given time span is out.

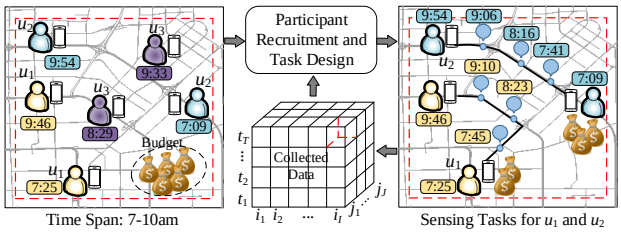


Figure 1. The procedure of our urban sensing framework.

Such an urban sensing framework can provide participants with unobstructed tasks that do not break their original com-

*The research was done when the first author was an intern in Microsoft Research under the supervision of the second author.
†Yu Zheng is the correspondence author of this paper.

UbiComp '16, September 12-16, 2016, Heidelberg, Germany
 ISBN 978-1-4503-4461-6/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971735

muting plans, while ensuring a sensing program has the best coverage of data that can better support upper-level applications (check *Related Work* for more details). To achieve this goal is challenging, however, given the following reasons: *First*, to define the coverage of collected data is non-trivial in a spatio-temporal space, where spatial and temporal dimensions can have different granularities [18]. Partitioned by different geographical sizes and different lengths of temporal intervals, collected data will present different distributions. The evaluation of data coverage will significantly affect the task design and participant recruitment. *Second*, while there are numerous task candidates that can be assigned to a participant (e.g. multiple paths satisfying its commuting plan), we try to find a near-optimal task to maximize the data coverage. *Third*, to recruit participants from a given pool of candidates is a NP-hard problem [16, 2], which is hard to solve efficiently.

To address these issues, we propose an urban sensing framework that is comprised of an objective function, a graph-based task design algorithm and a participant recruitment mechanism. Our contributions are four-fold:

- The objective function measures the coverage of collected data through computing the entropy of data represented with different spatio-temporal granularities (e.g. the hierarchy of geographical spaces and the length of time intervals). This function quantifies the coverage of data for an urban sensing program, improving the task design and participant recruitment significantly.
- The graph-based task design algorithm constructs a location graph based on a participant's mobility and the data that has been collected, finding a near-optimal task that maximizes the data coverage of the sensing program for the participant using a dynamic programming strategy.
- The participant recruitment mechanism first randomly selects participants from a candidate pool and then replaces these selected participants gradually by other candidates left in the pool, until the coverage of the data that will be potentially collected is not improved anymore. This mechanism solves the NP-hard problem efficiently, resulting in a data coverage close to those generated by computational expensive methods, such as the greedy-based mechanism.
- We evaluate our framework based on the real human mobility of 34 users. The field study and simulation results have shown the advantages of our framework beyond multiple baseline approaches.

OVERVIEW

Preliminary

Definition 1 *(Participant)*: A participant is a tuple $u =< l_s, l_e, t_s, t_e, r >$, where l_s, l_e, t_s, t_e are the participant's origin, destination, departure time and arrive time, respectively, and r refers to the reward we will give to the participant. r is calculated based on the participant's slack time:

$$r = r_t \times \{(t_e - t_s) - t(l_s, l_e)\} \tag{1}$$

where r_t is the unit reward per hour and the latter part is the slack time of participant u; $t(l_s, l_e)$ denotes the necessary travel time between l_s and l_e.

Definition 2 *(Task)*: An urban sensing task s is a sequence of collecting locations associated with collecting time intervals, in which data is to be collected, i.e., $s = (l_1, t_1) \rightarrow (l_2, t_2) \rightarrow \cdots$, where l denotes the collecting location and t refers to the collecting time interval.

Definition 3 *(Sensed Data)*: The sensed data is represented by a three-dimension tensor $\mathcal{A}$, where an entry $\mathcal{A}(i, j, t)$ denotes the amount of data collected in location (i, j) at time interval t. In this study, we divide a city into uniform grids. Location (i, j) denotes the grid at the i-th row and j-th column.

Problem Definition

Given a pool of participant candidates $U = \{u_1, u_2, ..., u_n\}$ and a budget B, the urban sensing problem is to 1) recruit a portion of participants U' in U, 2) design a task s_u for each recruited participant $u \in U'$, such that the coverage of data collected $\phi(\mathcal{A})$ is maximized, formally defined as:

$$\begin{aligned} &\max \quad \phi(\mathcal{A}) \\ &s.t. \quad \begin{cases} \sum_{u \in U'} u.r \leq B \\ \mathcal{A} = \sum_{u \in U'} \mathcal{A}_{s_u} \end{cases} \end{aligned} \tag{2}$$

where $\mathcal{A}_{s_u}$ is the data sensed by s_u; we will detail $\phi(\mathcal{A})$ later.

Framework

Figure 2 presents the framework of our urban sensing approach, which is comprised of three main components: join and sensing, task design, and participant recruitment.

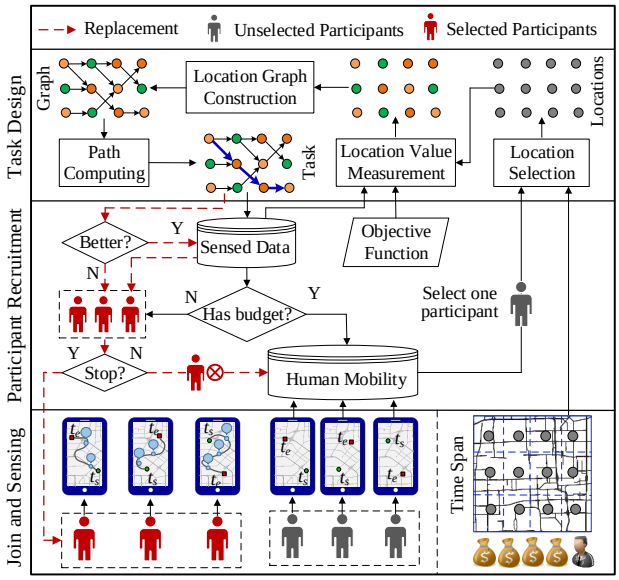


Figure 2. The framework of our urban sensing system.

In the first component, as illustrated in the bottom part of Figure 2, people can create an urban sensing program by submitting a geographical region and a time span where data will be collected, together with the type of data to be collected, a budget (e.g. the amount of money they would like to pay), and the minimum reward that a participant can obtain each hour. If interested, participants (denoted by gray human icons) can submit information on their mobility, consisting of an origin, a destination, a departure time, and an arrival time, through

their mobile devices. If being recruited, a participant (denoted by red human icons) will receive a task which consists of a sequence of collecting points and corresponding time intervals where the participant should collect data. Afterwards, those selected participants go collect data in the real world following the assigned task and return the collected data to our system through their mobile devices.

In the task design component, as depicted in the top of Figure 2, we design a task for each participant, based on their mobility and the data that is expected to be collected, through four steps. *First*, we check each location individually in the geographical region, finding the locations that can be reached by a participant between their departure time and arrival time. Each selected location is associated with a time interval in which data can be collected (called collecting time interval). *Second*, we measure the value of each selected location (for improving data coverage, denoted by different colors in Figure 2) based on an objective function and the data that is supposed to be collected. For example, if a location or its spatial neighbors have just been sensed by other participants, the value of data at this location should be lower than those that have never been explored. We call such values of data coverage value. In the *third* step, we connect two selected locations l_1 and l_2 if a participant can reach l_2 at its collecting time interval after collecting data in l_1. In this way, we can construct a location graph with each node denoting a location with a collecting time interval and a coverage value. In the location graph, each path from the participant's origin to destination is an unobstructed task candidate. Finally, we search the location graph for a near-optimal path with near-maximum coverage value.

The participant recruitment component is comprised of two steps: participant selection and participant replacement [2], as demonstrated in the middle part of Figure 2. First, we select participants from a candidate pool, randomly and one-by-one. By calling the task design component, we assign each participant a task and update the data that has been planned to be collected (denoted by the sensed data). At this moment, the data has not really been collected yet. The total budget is then reduced by the reward that will be given to the participant. We repeat the participant selection process until the budget has been used up. Afterwards, we start the participant replacement process, which randomly replaces one participant from the selected group (denoted by the red human icons) with another participant from the candidate pool. If the data coverage is improved by the replacement, we keep the change; otherwise, we drop the replacement and continue to find another pair of participants to do the replacement. We repeat the replacement process until the data coverage is not improved at all after certain times (e.g. 100 times) of consecutive attempts.

OBJECTIVE FUNCTION

In this section, we discuss the difficulties of evaluating data coverage and then we propose a hierarchical entropy-based objective function.

Difficulty on Evaluating Data Coverage

In an urban sensing application, data coverage is usually measured by its amount and distribution. While the amount of data is easy to measure, the distribution of data is difficult to evaluate in a spatio-temporal space. Partitioned by different geographical sizes and different lengths of temporal intervals, collected data will present different distributions. For example, as shown in Figure 3, with the given spatial and temporal space, we can have both a fine-grained partition (Figure 3E) and a coarse-grained partition (Figure 3F). With the fine-grained partition in the spatial space in Figure 3A and Figure 3C, the two datasets (denoted by blue circles) can both cover 4 out of 16 grids, which indicates that they may have the same data coverage. However, if it has a coarse-grained partition, the first dataset only covers 1 out of 4 grids (Figure 3B), while the second dataset can cover the entire 4 grids (Figure 3D).

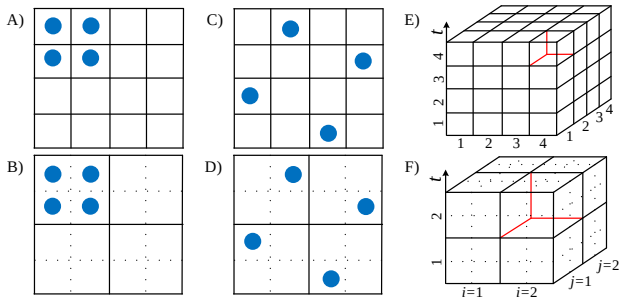


Figure 3. Different granularities of partitions and data distributions.

Hierarchical Entropy-based Objective Function

To address this issue, we propose a hierarchical entropy-based objective function $\phi(\mathcal{A})$, defined as

$$\phi(\mathcal{A}) = \alpha E(\mathcal{A}) + (1-\alpha)\log_2 Q(\mathcal{A}), \tag{3}$$

where $E(\mathcal{A})$ denotes the hierarchical entropy of the collected data $\mathcal{A}$, measuring the balance of data $\mathcal{A}$; $Q(\mathcal{A})$ refers to the amount (quantity) of the collected data; $\alpha \in [0,1]$ is a parameter to tune the importance of data balance to data amount, which can be application-specific. For example, we can set α larger to make an urban sensing program focus mainly on collecting balanced data. To make α more sensitive, in our objective function (Equation 3), we apply $\log_2 Q(\mathcal{A})$, instead of $Q(\mathcal{A})$, as $\log_2 Q(\mathcal{A})$ and $E(\mathcal{A})$ are with more similar order of magnitude. Formally, the data amount $Q(\mathcal{A})$ is

$$Q(\mathcal{A}) = \sum_{i,j,t} \mathcal{A}(i,j,t), \tag{4}$$

where $\mathcal{A}(i,j,t)$ denotes the amount of data collected in grid (i,j) at time interval t. Below, we detail the data balance $E(\mathcal{A})$.

To measure the data balance of the collected data $\mathcal{A}$, we propose the hierarchical entropy $E(\mathcal{A})$ to simultaneously consider the data balance at different spatial and temporal granularities (i.e., from fine-grained partitions to coarse-grained partitions). The insight is that a balanced data coverage is expected to be balanced at each granularity. Specifically, the data balance at each granularity is evaluated using entropy and a balanced data distribution has a large entropy. That is, our hierarchical data balance $E(\mathcal{A})$ is the entropy aggregation of data distribution at each granularity. We define it as

$$E(\mathcal{A}) = \sum_{k=1}^{k_{max}} \omega(k) E(\mathcal{A}(k))/k_{max}, \tag{5}$$

where k_{max} is the number of granularities we consider in the given spatio-temporal space; the weight factor $\omega(k)$ is to make data balance at each granularity contribute equally to $E(\mathcal{A})$; $E(\mathcal{A}(k))$ is the data balance (i.e., the entropy) at granularity k, which is defined as:

$$E(\mathcal{A}(k)) = -\sum_{i,j,t} p(i,j,t|k)\log_2(p(i,j,t|k)), \tag{6}$$

where $p(i,j,t|k) = \mathcal{A}(i,j,t|k)/Q(\mathcal{A})$; $\mathcal{A}(i,j,t|k)$ is the amount of data collected in grid (i,j) at interval t from the perspective of granularity k. For granularity k, we denote $I(k), J(k)$ as the number of rows and columns in the spatial space and $T(k)$ as the number of time intervals. When the data $\mathcal{A}$ is evenly distributed at granularity k, we can get the maximum entropy $\log_2 I(k)J(k)T(k)$. Consequently, we can set the weight factor $\omega(k)$ in Equation 5 as $\omega(k) = \frac{\log_2 I(1)J(1)T(1)}{\log_2 I(k)J(k)T(k)}$ such that for each granularity k, $\omega(k)E(\mathcal{A}(k))$ has the same maximum value.

To demonstrate the effectiveness of the proposed hierarchical entropy $E(\mathcal{A})$, as shown in Figure 4, we consider two datasets in the spatial space. First, we have $k_{max} = 2$ granularities of partitions with $I(1) = J(1) = 4$ and $I(2) = J(2) = 2$. Then we can get $E(\mathcal{A}) = 1$ for the first dataset (Figure 4A) and $E(\mathcal{A}) = 3$ for the second (Figure 4B). That's, the dataset in Figure 4B is much more balanced than the dataset in Figure 4A, which is quite consistent with our common knowledge.

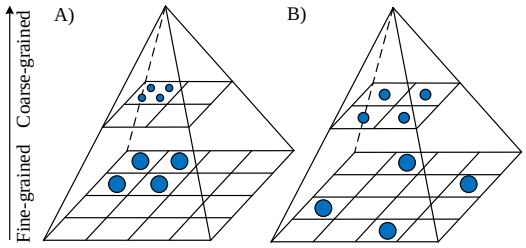


Figure 4. Data balance considering hierarchical partitions.

Given the collected data $\mathcal{A}$, to efficiently compute the objective value $\phi(\mathcal{A})$, we provide a method which takes only $O(1)$ time complexity. The method is detailed in Appendix 1. In comparison, if without the efficient method, computing $\phi(\mathcal{A})$ directly based on its definition is with $O\{\sum_{k=1}^{k_{max}} I(k)J(k)T(k)\}$ time complexity, which can be fairly expensive for two aspects. On one hand, as we'll see later, we need to compute the objective value $\phi(\mathcal{A})$ very frequently when designing tasks for participants. On the other hand, $I(k), J(k), T(k)$ can be very large in some real-world urban sensing applications.

TASK DESIGN

Location Selection

In our urban sensing framework, an urban sensing task is defined as a sequence of collecting locations with corresponding time intervals. Thus, before designing such a location sequence for a participant, we need to first select all individual locations that the participant can reach, based on its mobility information. Note that a location can be associated with many time intervals, as participants may sense data in a location at different time intervals. Similarly, for a time interval, there could be many locations that the participant can reach.

To select all such individual locations, we check each location in the given geographical region. Specifically, if a participant can reach a location at a time interval while not violating its original commuting plan, two constraints should be satisfied: 1) before the end of this time interval, the participant can reach the location and finish sensing the data; 2) after sensing the data, the participant can reach its destination timely (i.e., before the given arrival time) to not break its original commuting plan. Basically, it takes some time to perform data sensing.

Obviously, participants' travel time estimation is crucial. For a given commuting mode (e.g. driving), travel time estimation is easy with previously well studied methods. For example, using the method proposed in [19], taxicabs' travel time estimation error is just 2.6 minutes on average. Actually, we can also put aside some slack time for potential unexpected situations (e.g. traffic accidents). However, in some cases, participants will take two or more commuting vehicles during one single trip (e.g. driving → bicycle → bus) and travel time estimation becomes complicated: 1) Participants can clearly report each commuting mode's origin, destination, departure time, and arrival time, then we can split a trip with many commuting modes into many sub-trips with one commuting mode [23], whose travel time can be well estimated; 2) Otherwise, travel time estimation is difficult (beyond our consideration scope).

We detail the process of location selection using the example in Figure 5. A time interval is supposed to be 10 minutes. To check a location (location 2 for example), we need to go through two steps. First, we estimate the travel time needed for the participant traveling from its origin to location 2 (18 minutes) and the travel time from location 2 to its destination (16 minutes), as shown in Figure 5A. It implies that the participant can reach location 2 at 8:43 and should leave for destination before 8:58. Next, we find that location 2 can only be associated with two time intervals, i.e., 9:40-50 and 9:50-60 (Figure 5A), ruling out the time intervals before 8:43 (Figure 5B) and after 8:58 (Figure 5C). In this way, we can check each location and obtain all individual locations that the participant can reach. Let's denote L_t as the set of locations that the participant can reach at time interval t. In this example, we have $L_{8:40-50} = \{1,2,5\}$ (Figure 5D), $L_{8:50-60} = \{2,5,6\}$ (Figure 5E) and $L_{9:00-10} = \{3,6\}$ (Figure 5F).

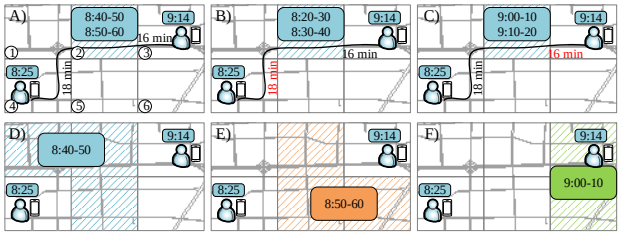


Figure 5. Location selection for a participant.

Location Value Measurement

Given the current collected data $\mathcal{A}$, sensing data in different locations may have different contributions to overall data coverage $\phi(\mathcal{A})$. We call such a contribution of a location the *coverage value*. Intuitively, a location with more coverage value is more likely to be included in a designed task. The coverage value of a location is also associated with time intervals

as a location at different time intervals could contain different coverage value. Formally, given the already sensed data $\mathcal{A}$, the coverage value of the location $l = (i, j)$ at time interval t, denoted by $\phi\{(l, t)|\mathcal{A}\}$, is formulated as

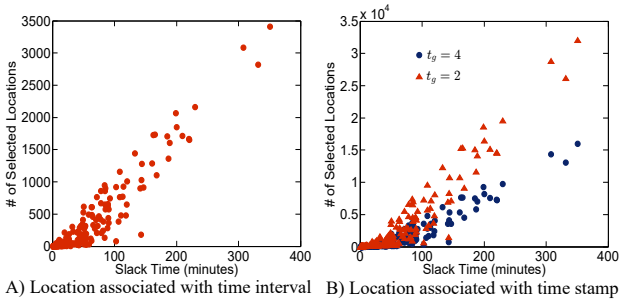
$$\phi\{(l, t)|\mathcal{A}\} = \phi(\mathcal{A} + \mathcal{A}_{(l,t)}) - \phi(\mathcal{A}), \tag{7}$$

where $\mathcal{A}_{(l,t)}$ refers to the data sensed at location $l = (i, j)$ at time interval t, i.e., $\mathcal{A}_{(l,t)}$ is null except $\mathcal{A}_{(l,t)}(i, j, t) = 1$. Indeed, the coverage value $\phi\{(l, t)|\mathcal{A}\}$ is the marginal coverage gain if new data in location l at time interval t is designed to be added into the already sensed data $\mathcal{A}$.

As we expect a balanced data coverage in both the spatial and temporal space, for the coverage value of a location there are two typical cases: 1) If some participants have just sensed data in a location or its spatial neighbors, the coverage value in this location at current time interval will drop significantly; 2) On the contrary, a location that has not been explored for a long time, is supposed to be with more coverage value.

Location Graph Construction

With selected individual locations, to extract all task candidates is definitely non-trivial. As defined, a task candidate is a location sequence $(l_1, t_1) \rightarrow (l_2, t_2) \rightarrow \cdots$, which satisfies $t_1 \leq t_2 \leq \cdots$. However, not all location sequences satisfying the constraint is a task candidate. Intuitively, even if $t_1 \leq t_2$, the participant still might not be able to reach l_2 from l_1 within $t_2 - t_1$, especially when l_2 and l_1 are far away from each other. This means we need to further check whether a location sequence is a valid task candidate or not, based on the participant's travel time needed between two locations. However, to check all possible location sequences is not possible, due to its extremely high cardinality. Specifically, for a participant with selected individual locations at different time intervals, denoted by $L_1, L_2, \cdots, L_{t_{max}}$, there are totally $\prod_{t=1}^{t_{max}} \sum_{m=1}^{|L_t|} \frac{|L_t|!}{||L_t|-m|!}$ location sequences. Obviously, the number of location sequences grows exponentially to the amount of individual locations. In the real world, unfortunately, there could be hundreds of individual locations, especially when participants have lots of slack time (Figure 6A).



A) Location associated with time interval B) Location associated with time stamp

Figure 6. Real-world statistics on the cardinality of selected locations.

To efficiently extract task candidates, we construct a location graph (Figure 9) for a participant, in which vertices include all selected individual locations, the participant's origin and destination. We study the spatio-temporal relation between any two selected individual locations, and if the participant can reach location l_2 at time interval t_2 from location l_1 at t_1 (i.e., reach location (l_2, t_2) from location (l_1, t_1)), we link a directed edge from location (l_1, t_1) to location (l_2, t_2) (Figure 8). Hence in the location graph a path from the participant's origin to destination, implying the participant is able to cover each location one by one, is a task candidate, and to extract all (unobstructed) task candidates is to extract all such paths.

However, to construct such a location graph is not easy, since sometimes we can't identify whether the participant can reach location (l_2, t_2) from location (l_1, t_1). For example, as shown in Figure 7A, if we only know the participant reaches location 5 at time interval 8:40-50, there could be two different situations: 1) the participant reaches location 5 at 8:41 (Figure 7B) and then she can reach 2 at 8:50-60; 2) the participant reaches location 5 at 8:45 (Figure 7C) and will fail to reach location 2 at 8:50-60. The key point here is that, the participant may reach the location at any time stamp in the given time interval, which will lead to completely different results.

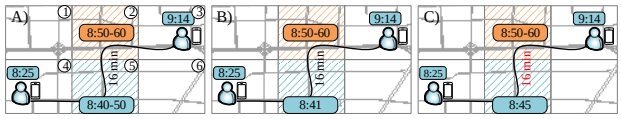


Figure 7. Importance of time stamp information for graph construction.

To address this issue, we intend to select locations that a participant can reach before some exact time stamps, i.e., some specific time stamps in the time intervals. As illustrated in Figure 8, given the mobility information of the participant, we first select some *specific time stamps* (e.g. 9:00, 9:04 and 9:08). And then we select the locations that the participant can reach before each specific time stamp, that's $L_{9:00} = \{1, 2, 3, 5, 6\}$, $L_{9:04} = \{3, 4, 6, 7, 9, a\}$ and $L_{9:08} = \{7, 8, b, c\}$. The specific time stamps can be set to be with an equal time gap t_g (e.g. $t_g = 4$ minutes in Figure 8) and when t_g approaches 0, the locations associated with time stamps are close to those associated with time intervals. Based on the exact time stamps and the estimated travel time between locations, we can directly tell whether a participant is able to reach one location from the previous one, and if yes, there is an edge (see Figure 8).

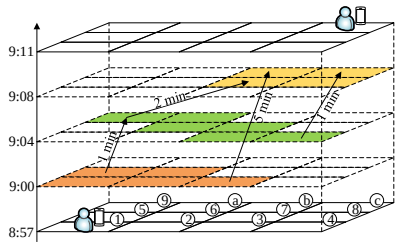


Figure 8. Location selection associated with specific time stamps, i.e., 9:00, 9:04 and 9:08 ($t_g = 4$).

Now, we can formally construct a location graph $G(u|\mathcal{A})$ for a participant u, as shown in Figure 9. The vertices V in the graph $G(u|\mathcal{A})$ contains two parts: 1) each selected location l with a time stamp t and a coverage value as defined in Equation 7 (for a location, the coverage value at a time stamp is its coverage value at its corresponding time interval); 2) the origin l_s and destination l_e of the participant u associated with its departure time t_s and arrival time t_e, respectively. The edges E in the location graph is also comprised of two components: 1) according to the location selection process, there always exists

a directed edge from the participant's origin (l_s, t_s) to each selected location (l, t) and a directed edge from each selected location (l, t) to the participant's destination (l_e, t_e); 2) for any two locations (l_1, t_1) and (l_2, t_2), if the participant can reach l_2 before the time stamp t_2 from l_1 at time stamp t_1, then there is a directed edge from location (l_1, t_1) to location (l_2, t_2).

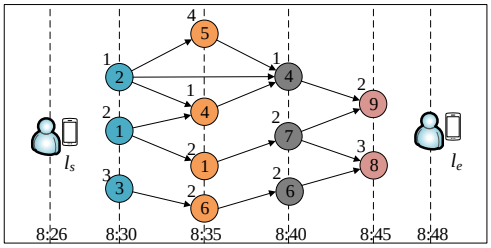


Figure 9. The constructed location graph for a participant. The figure around each location denotes its value, for example, the coverage value of location 3 at 8:30 is 3. The first type of edges are not demonstrated.

Constructing such a location graph has little influence on the time efficiency of our task design, given the following two reasons: 1) we can construct the location graph once receiving the participant's mobility information, before the urban sensing program starts; 2) given different $\mathcal{A}$'s, we just need to update the coverage value for each location (Equation 7), leaving the remaining parts of the location graph unchanged.

Path Computing

Based on the location graph, we can give a formal definition for task design now: to design the best task s_u for participant u is to find the optimal path from the participant's origin (l_s, t_s) to destination (l_e, t_e), denoted by $(l_s, t_s) \rightarrow (l_1, t_1) \rightarrow (l_2, t_2) \rightarrow \cdots \rightarrow (l_M, t_M) \rightarrow (l_e, t_e)$, such that the overall coverage value of all locations in task $s_u = (l_1, t_1) \rightarrow (l_2, t_2) \rightarrow \cdots \rightarrow (l_M, t_M)$ is maximized. Mathematically, it is

$$\max \quad \phi(s_u|\mathcal{A}) = \phi(\mathcal{A} + \mathcal{A}_{s_u}) - \phi(\mathcal{A}), \tag{8}$$

where $\phi(s_u|\mathcal{A})$ refers to the marginal improvements for data coverage if all data in the task s_u is designed to be sensed; $\mathcal{A}_{s_u}$ denotes the data sensed from the task s_u.

Nonetheless, to find such an optimal path could still be very computationally expensive. As our hierarchical entropy-based objective function ϕ is non-linear, i.e., $\phi(s_u|\mathcal{A}) \neq \sum_{m=1}^{M} \phi\{(l_m, t_m)|\mathcal{A}\}$, to get the optimal path we must check each path from the participant's origin to destination, whose cardinality can be quite large. As shown in Figure 6B, in real-world scenarios, the number of selected individual locations (vertices for the graph) grows drastically with the increase of the participant's slack time, which could result in numerous paths from the participant's origin to destination.

To address this issue, we intend to find an efficient near-optimal path (task) such that only a very small portion of paths from the participant's origin (l_s, t_s) to destination (l_e, t_e) are needed to be tested. Specifically, instead of maximizing the overall coverage value of all locations in a path (Equation 8), we turn to maximize the sum of each location's coverage value, i.e.,

$$\max \quad \sum_{m=1}^{M} \phi\{(l_m, t_m)|\mathcal{A}\}. \tag{9}$$

To efficiently find such a near-optimal path (task), we provide a *dynamic programming* strategy [4] based on two observations. *First*, as shown in Figure 9, the location graph is directed and acyclic such that selected locations can be classified into different layers according to their time stamps, denoted by $L_1, L_2, \cdots$ (e.g. $L_1 = \{2, 1, 3\}$ in Figure 9). Edges start from locations with smaller time stamps (called lower layers) to higher layers. *Second*, based on the first observation, we can iteratively compute the best path from l_s to each location l, layer by layer. Initially, the best path from l_s to each location l at the first layer L_1 is directly to be $(l_s, t_s) \rightarrow (l, t)$. To find the best path from l_s to each location l at the second layer L_2, taking (4,8:35) for example, we first find all locations linking to (4,8:35), denoted by $(4,8:35)^{\text{in}}$={(2,8:30),(1,8:30)}. Next, we select the best vertex from $(4,8:35)^{\text{in}}$, i.e., (1,8:30), as the best path $(l_s, t_s) \rightarrow$(1,8:30) (with coverage value 2) contains more coverage value than the best path $(l_s, t_s) \rightarrow$(2,8:30) (with coverage value 1). Finally, the best path from l_s to (4,8:35) is directly to be $(l_s, t_s) \rightarrow$(1,8:30)$\rightarrow$(4,8:35). In this way, we can iteratively compute the best path from (l_s, t_s) to each selected location (l, t), including (l_e, t_e), from which we get the task s_u. We formally detail our dynamic programming process below.

1. For each $l \in L_1$, the best path from l_s to l is $Path\{(l_s, t_s) \rightarrow (l, t)\} = (l_s, t_s) \rightarrow (l, t)$ and the coverage value of each location in the path is $\phi\{(l_s, t_s) \rightarrow (l, t)\} = \phi\{(l, t)|\mathcal{A}\}$.
2. For each $l \in L_t$ $(t \geq 2)$, we first extract those locations which have a link to (l, t), which is denoted by $(l, t)^{\text{in}}$. For example, in Figure 9, we have $(4,8:40)^{\text{in}}$ ={(2,8:30),(4,8:35),(5,8:35)}. Then, we find the best location (l_{best}, t_{best}) from $(l, t)^{\text{in}}$ where $Path\{(l_s, t_s) \rightarrow (l_{best}, t_{best})\}$ is with most coverage value, i.e., $(l_{best}, t_{best}) = arg\max_{(l_0,t_0)\in(l,t)^{\text{in}}} \phi\{(l_s, t_s) \rightarrow (l_0, t_0)\}$. After that, we directly obtain the best path $Path\{(l_s, t_s) \rightarrow (l, t)\} = Path\{(l_s, t_s) \rightarrow (l_{best}, t_{best})\} \rightarrow (l, t)$ with the coverage value $\phi\{(l_s, t_s) \rightarrow (l, t)\} = \phi\{(l_s, t_s) \rightarrow (l_{best}, t_{best})\} + \phi\{(l, t)|\mathcal{A}\}$.
3. Finally, we get the task s_u from $Path\{(l_s, t_s) \rightarrow (l_e, t_e)\}$.

To help participants fulfill the designed tasks, we have two useful mechanisms in our framework. First, in our definition, a task is a sequence of sensing points (locations) rather than fixed paths. Participants can flexibly choose their own path between two consecutive sensing points, just making sure each assigned location can be well sensed. Second, each location in a task is associated with a time interval rather than a time stamp, i.e., participants have more flexibilities. For example, assume the location sequence (2,8:30)$\rightarrow$(4,8:40)$\rightarrow$(9,8:45) is the task extracted using our algorithm, then before assigning it to a participant, we will transfer it to (2,8:30-40)$\rightarrow$(4,8:40-50)$\rightarrow$(9,8:40-50) if a time interval is set to be 10 minutes.

Even though a task might not be well finished by a participant, we have an error tolerance mechanism. There do exist some unexpected situations that will fail designed tasks, such as urgent things happening to a participant and its original commuting plan needing to be changed temporarily. To address this issue, we can re-conduct the participant recruitment and task design for the participants whose departure times are after this unexpected failed task. That is, we can dynamically adjust the recruitment and task design based on the actual data collected by previously recruited participants.

EVALUATION

This section evaluates our urban sensing framework using the real human mobility of 34 participants which is extracted from a real-world noise sensing experiment.

Datasets

We first briefly introduce the real-world noise sensing experiment, from which we extract real human mobility datasets for evaluating our framework. The noise sensing experiment is conducted in a $6.6km \times 3km$ geographical region in Beijing, from May 15-20 and from June 9-19, 2015. For each day, the sensing time span is from 6:00 am to 22:00 pm. 34 college students are recruited as participants to sense noise data using built noise sensing Apps. In this experiment, instead of designing tasks for participants, we let participants make decisions on where and when to sense data based on their real commuting plans. It takes 2 minutes for participants to sense noise data, and the more data they collect, the more rewards they will get (each data is with some rewards). Thus to earn more rewards participants need to put more slack time. When participants conduct noise sensing, their GPS trajectories will be recorded every 30 seconds through the App and will be sent to our back-end server later. In this real-world experiment, we spend 8000 *RMB* with 3313 noise data collected. As shown in Figure 10C, without well designed tasks for participants in this experiment, the collected noise data is highly skewed in the geographical space, i.e., redundant data in areas nearby the participants' colleges and scarce data in remaining areas.

From the recorded GPS trajectories of participants, we can extract their mobility information (commuting plans). Specifically, given a trajectory in which the participant has consecutive sensing behavior (if not, a trajectory will be split into sub-trajectories), the location and time stamp of the first sensed data are deemed as the origin and departure time of the participant respectively, and similarly the location and time stamp of the last data as its destination and arrive time respectively. In some cases, both the origin and destination of participants (students) are their colleges, that is students spend some time just for joining our sensing experiment to earn some rewards. Also, we can easily extract the moving velocity of each participant from the recorded trajectories. As participants (students) usually go to sense data on foot or by riding a bicycle, which are not sensitive to traffic conditions, we can directly use the extracted moving speed to conduct travel time estimation (i.e., road distance divided by speed). And then we can estimate participants' slack times (see Equation 1).

Overall we extract 244 records of commuting plans and after integrating them into one day, we have 244 participant candidates with mobility information to evaluate our urban sensing framework. As depicted in Figure 10A and 10B, the spatial distribution of participants' commuting plans are highly skewed, i.e., redundant human coverage nearby participants' colleges and little human coverage in other areas. We expect our urban sensing framework can get a good data coverage even with such a skewed human mobility dataset.

Experiment Settings

To evaluate our urban sensing framework, we first deem the previously extracted 244 participants with human mobility as

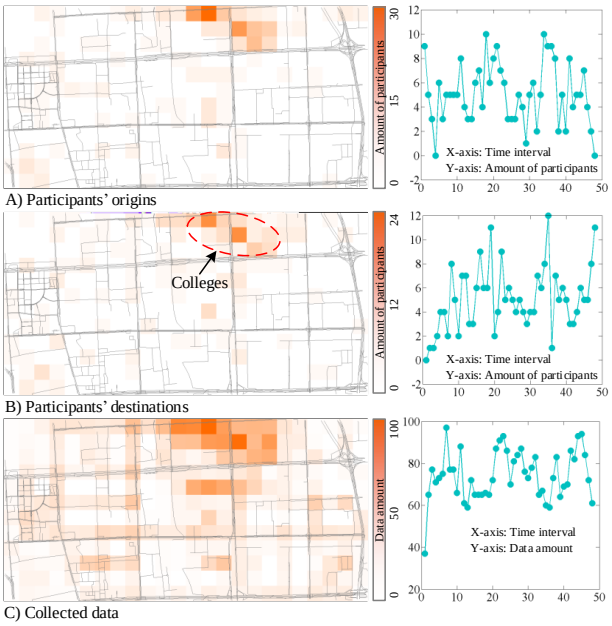


Figure 10. Spatial and temporal distribution of participants' origin (A), destination (B) and collected data (C).

the participant candidates who submit their mobility information to our system, based on which we conduct the participant recruitment and task design in our framework. As the human mobility is real, we believe our experiment results can well guide real-world implementations of our urban sensing system in the future. In our evaluation, the sensing region and the time span keep the same with those in the previous noise sensing experiment. We partition the geographical region into 24×12 grids and the sensing time span into 48 time intervals, based on which we design sensing tasks to participants. To model our hierarchical entropy for data balance (Equation 5), we consider another 3 granularities of partitions (i.e., $k_{max} = 3$) with parameters for each granularity summarized in Table 1. In addition, if without special clarification, we have the following settings by default: 1) the budget (i.e., money) is 5000 *RMB* and the unit reward per hour to participants is 40 *RMB*; 2) in task design, the specific time stamps that locations are associated with are set to be every 4 minutes, i.e., $t_g = 4$; 3) in participant recruitment, replacement algorithm will stop if no improvement can be reached after 100 times of consecutive attempts; 4) in objective function (Equation 3), we set α as 0.5, i.e., both data amount and data balance are important.

Granularity k	$I(k)$	$J(k)$	$T(k)$
1	12	12	24
2	8	6	12
3	4	3	6

Table 1. Parameters for different granularities of partitions.

Evaluation on Collecting Balanced Data

To support upper-level applications, a balanced data coverage is very important in many urban sensing scenarios, which, however, is also very challenging as the human mobility in a city can be highly skewed. To this end, we first evaluate our urban sensing framework on its capability for collecting data with a good coverage using the highly skewed human mobility

of participant candidates (see Figure 10A and Figure 10B). In our hierarchical entropy-based objective function (Equation 3), the parameter α tunes the importance of data balance to data amount. Thus we study three different cases here: 1) $\alpha = 0$: aiming at more data amount; 2) $\alpha = 0.5$: aiming at both data amount and data balance; 3) $\alpha = 1$: aiming at more balanced data. Note that the α can be application-specific in real life.

Figure 11 visually demonstrates the spatial and temporal distribution of the data collected by our framework. Compared with the highly skewed human mobility of the participant candidates (Figure 10A and Figure 10B), we surprisingly collect balanced data in the spatial space, especially when $\alpha = 1$ as shown in the Figure 11C. In fact, to collect more balanced data ($\alpha = 1$), participants are, to maximum extent, designed to cover the entire sensing region, especially for those participants with a lot of slack time (not breaking their original commuting plans). That is, we have provided a practical urban sensing framework to collect data with a balanced coverage even using the intrinsically skewed human mobility in real life. As we can't modify participants' departure times and arrival times (we provide unobstructed tasks), there remains little room to improve data balance in the temporal space.

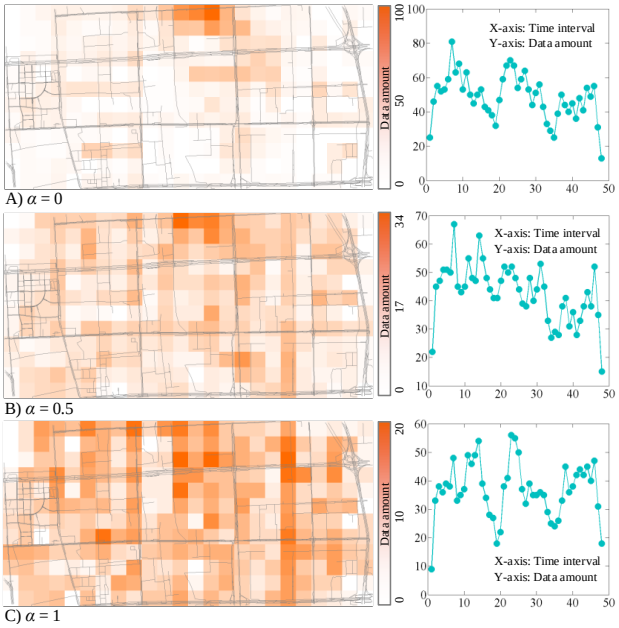


Figure 11. Spatial and temporal distributions of collected data using our framework with different α.

In addition to the visual results, we obtain some statistics presented in Figure 12. We are capable of receiving more data amount ($\alpha = 0$) or better data balance ($\alpha = 1$) while the case $\alpha = 0.5$ provides a good trade-off between the data amount and the data balance. As depicted in Figure 12, as the α increases, the entropy of the collected data grows while the amount of the data decreases, which agrees with our inner setting for the α. It indicates that apart from the data balance, our framework can also maximize the data amount by setting $\alpha = 0$. In essence, the task design mechanism in our framework endows participants with higher sensing capability to cover more geographical space, no matter what the specific objective function is. Therefore we believe our task design framework can be well applied to different kinds of objective functions in real-world urban sensing scenarios.

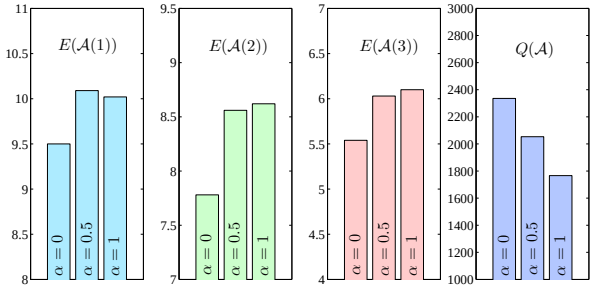


Figure 12. Performance comparison between different α.

Evaluation on the Hierarchical Entropy

In this subsection, we study whether our proposed hierarchical entropy can measure data balance accurately in the spatio-temporal space. To this end, we set the hierarchical entropy as our objective function, i.e., $\phi(\mathcal{A}) = E(\mathcal{A})$. We compare it with baselines, which, by contrast, consider data balance (entropy) only at one single granularity. Therefore, we have three baselines for comparison, including the data balance at the first granularity $\phi_1(\mathcal{A}) = E(\mathcal{A}(1))$, the second granularity $\phi_2(\mathcal{A}) = E(\mathcal{A}(2))$ and the third granularity $\phi_3(\mathcal{A}) = E(\mathcal{A}(3))$.

As shown in Table 2, each baseline tends to collect balanced data (large entropy) only at its own granularity. For example, $\phi_3(\mathcal{A})$ has the most entropy at the third granularity, with, unfortunately, the least entropy for both the first and second granularity. As a balanced data coverage is expected to be balanced at each granularity, data balance considering only one single granularity (e.g. $\phi_1(\mathcal{A})$, $\phi_2(\mathcal{A})$, $\phi_3(\mathcal{A})$) can be very biased. On the contrary, our hierarchical data balance $\phi(\mathcal{A})$ achieves good results at all granularities, the second best for each granularity. Also, our hierarchical data balance achieves the best overall performance $E(\mathcal{A})$. That is, our hierarchical entropy can best measure spatio-temporal data balance.

	$E(\mathcal{A}(1))$	$E(\mathcal{A}(3))$	$E(\mathcal{A}(3))$	$E(\mathcal{A})$
$\phi(\mathcal{A})$	10.02	8.61	6.10	**10.90**
$\phi_1(\mathcal{A})$	**10.16**	8.35	5.82	10.64
$\phi_2(\mathcal{A})$	9.87	**8.68**	6.03	10.84
$\phi_3(\mathcal{A})$	9.44	7.97	**6.14**	10.45

Table 2. Comparison between hierarchical entropy and baselines.

Evaluation on Time Efficiency

Besides effectiveness, an urban sensing application also requires high time efficiency. Here, we study the time efficiency of our urban sensing framework. Our experiments are performed using C# programs on a server with 2.40 GHz Intel(R) Xeon(R) CPUs. From the experiment results, our dynamic programming strategy can design a task within 2 seconds on average. The efficient task design can be attributed to two aspects as discussed previously: 1) we can construct the location graph (14 seconds on average) for each participant just after the participant submits its commuting plan, i.e., it won't affect the efficiency of our task design; 2) a very efficient method (see Appendix 1) is proposed to compute the coverage value for each selected location in our task design. Based on the efficient task design algorithm, only less than 10 minutes are necessary for the participant recruitment mechanism.

Evaluation on Participant Recruitment Mechanism

This subsection investigates the performance of our participant recruitment mechanism, which is comprised of a random algorithm and a replacement algorithm. For a better evaluation, we compare our mechanism with the following two baselines.

- Random recruitment: As the first step of our mechanism, the random recruitment doesn't use the replacement algorithm to further refine its recruitment performance.
- Greedy recruitment: Instead of selecting participants in a random manner, each time the greedy algorithm selects the best participant who marginally contributes most to the overall data coverage. To find such a participant, each time the greedy algorithm needs to conduct task design for each unselected participant, taking a lot of running time.

Table 3 presents the experiment results where we compare the three mechanisms from two aspects. *First*, for data coverage, our participant mechanism defeats the two baselines in terms of both data balance and data amount. The random recruitment, however, gets the worst performance. *Second*, for running time, our mechanism is very efficient (8 minutes), taking just more 3 minutes than the random algorithm. The greedy mechanism however is very time-consuming (55 minutes) because it needs to do much more times of task design as mentioned earlier. As expected, the replacement mechanism in our framework can efficiently and effectively improve the performance of the random mechanism. Compared with this two baselines, our mechanism is the best.

	$E(\mathcal{A}(1))$	$E(\mathcal{A}(2))$	$E(\mathcal{A}(3))$	$Q(\mathcal{A})$	Running Time
Random	9.84	8.35	5.93	1780	$\approx$ 5 minutes
Greedy	9.94	8.46	6.04	1847	$\approx$ 55 minutes
Ours	10.08	8.56	6.05	2053	$\approx$ 8 minutes

Table 3. Comparison between our recruitment method and baselines.

Evaluation on Task Design

In our task design mechanism, the selected locations are associated with time stamps which are set to be every t_g minutes (e.g. $t_g = 4$). With smaller t_g, more individual locations can be selected (see Figure 6) leading to more task candidates being extracted, thus we can also have better results for both data balance and data amount as shown in Figure 13. However, smaller t_g also takes more running time for task design. Specifically, when $t_g = 4$, it takes just around 2 seconds to design a task, but when $t_g = 2$, it needs 8 seconds. In the real world, the program owner needs to make a trade-off between performance and running time when selecting t_g.

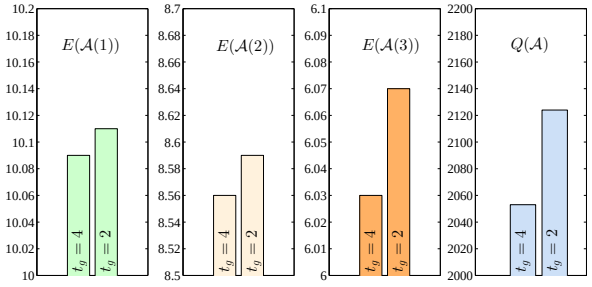


Figure 13. Comparison between different specific time stamps t_g.

RELATED WORK

Urban Data Evaluation Method

Data coverage ratio, the ratio of data collected across the whole sensing region and the entire sensing time span, is commonly used to measure the value of data collected. In general, a coverage ratio is predetermined before an urban sensing program starts, and the objective is to minimize the total rewards (i.e., budgets) paid to recruited participants while ensuring the collected data can meet the predetermined ratio [1, 7, 20]. Besides data coverage ratio, the value of data can also be specified by urban sensing program owners or experts [14]. And more recently, inferring the missing data using the already collected data, has been studied in [17, 22]. In this case, the collected data is deemed to be valuable if the missing data can be well inferred using the collected data.

In our work, we consider an urban sensing problem with a given budget, which could be more universal in real life. Accordingly we propose the hierarchical entropy-based data coverage evaluation method. Note that to predetermine a proper data coverage ratio is difficult when a budget is given. The proposed data evaluation method directly measures data balance and data amount by considering data distribution at different granularities of spatial and temporal space. We believe it could be a good data coverage evaluator, especially for urban sensing programs with a given budget constraint.

Task Design Based on Human Mobility

In many previous works [20, 1, 7], recruited participants collect data following their natural traveling paths (usually the quickest path from an origin to a destination). That is, participants collect data based on their natural mobilities, not designed by the program owner. To model participants' traveling paths, they use participants' historical trajectories [20] or mathematical models (e.g. Truncated Levy Walk model in [7] and Markov chain model in [1]). However, natural human mobility is intrinsically skewed (imbalanced) in the urban area, which has been well demonstrated by a large-scale real-world study in [3]. Hence, the performance of urban sensing based on participants' intrinsically skewed mobility is a big concern. For example, as shown in Figure 14, participants usually travel the quickest path (denoted by the orange path in A) from home to work (i.e., from a dwelling district to a business district). Areas along the quickest path contain redundant human coverage while the remaining areas are not covered at all. As a result, as depicted in Figure 14B, previous works only collect data along the quickest path, leaving most other areas not explored at all, resulting in a highly imbalanced data coverage.

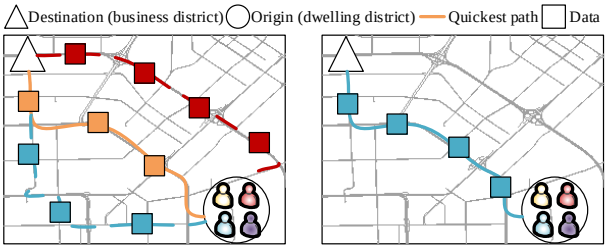


Figure 14. Tasks in our framework (A) and previous frameworks (B).

To deal with the skewed human mobility, [9] designs a reward mechanism to set discriminative rewards for data in different areas, aiming to steer participants to collect data in areas with little human coverage. An accurate model on participants' reactions to the discriminative rewards is critical to this framework. Such a model, however, could be very difficult given the complex related factors in the real world, which may include participants' psychologies, distribution of rewards, type of data to collect, weather condition, traffic condition, *etc.*

In this work, we propose a novel task design mechanism which can endow participants with higher sensing ability. Specifically, participants can be designed to sense those areas with little human coverage while containing valuable data (e.g. data with more coverage value, more specified utilities or more inference capabilities). The insight is that with some incentives (rewards), participants would be willing to 1) report their commuting plans and 2) travel new paths (represented by the light blue and red paths in Figure 14A) to reach their destinations as long as the arrival time is well controlled, e.g. before the working time in this example. The promotion of participants' sensing ability can provide a win-win solution for both urban sensing program owners and participants: 1) For program owners, they can collect more valuable data (more balanced coverage as shown in Figure 14A); 2) Participants are more likely to be recruited with rewards; In Figure 14, our framework recruits three participants while in the previous frameworks only one participant is necessary to avoid collecting redundant data.

Urban Sensing Applications

With the proliferation of rich-sensor mobile devices (e.g. smart phones, wearable devices), a number of urban sensing applications have been proposed [12, 11, 26, 25, 5, 13]. For example, in literature [12], Ouyang *et al.* propose a crowdsourcing-based event localization system to detect the location of events occurring in a city. The work [11] leverages the mobile crowdsensing to provide a weather estimation method in urban areas. Zheng *et al.* [25] propose a novel method to infer New York noise distribution using the 311 data collected by massive citizens, along with other sensed data. In [15], taxicabs are used to sense gas consumption and pollution emissions in a city.

Clearly, most of these applications need data with a balanced coverage to better deal with their problems. Our urban sensing framework is just designed to collect data with a good coverage even using intrinsically skewed human mobility in real life. Hence we believe our framework can well support these upper-level applications and will have broad applications.

CONCLUSION AND FUTURE WORK

In this paper, we provide a novel urban sensing framework based on real-world human mobility to maximize the coverage of data collected. One of the main results is that, through well designed tasks we can largely improve an urban sensing program's performance (e.g. data coverage), which was hampered by highly skewed human mobility in real life. Specifically, in our framework participants can be designed to sense areas with little human coverage but containing valuable urban data, yet not violating their original commuting plans. Another important result lies in the proposed hierarchical entropy-based data coverage evaluator. Our evaluator directly measures both data amount and data balance by considering different granularities of spatial and temporal space. We evaluate our framework using human mobility from the real world, with extensive experiments demonstrating its advantages over many baselines.

In the future, we plan to focus on two directions. First, participants may join our system after an urban sensing program starts [10, 8] such that both the task design and participant recruitment should be performed in real-time. Second, participants can also submit their expected rewards together with their commuting plans, making a truthful task design and participant recruitment mechanism necessary [2, 16].

ACKNOWLEDGEMENTS

This work was supported by the China National Basic Research Program (973 Program, No. 2015CB352400) and NSFC under grants No. U1401258 and No. 61573292.

APPENDIX

Appendix 1: Efficient Method to Get Objective Value

Given the sensed data $\mathcal{A}$, to efficiently compute the objective $\phi(\mathcal{A})$ (Equation 3), we propose an efficient method which takes $O(1)$ time complexity. Based on this method, in task design, we can also efficiently update each location's coverage value (Equation 7). To compute $\phi(\mathcal{A})$ is to compute $Q(\mathcal{A})$ and $E(\mathcal{A}(k))$, thus we just consider $Q(\mathcal{A})$ and $E(\mathcal{A}(k))$ below.

Assume we already know the current $\mathcal{A}$, $Q(\mathcal{A})$ and $E(\mathcal{A}(k))$, we intend to update the $Q(\mathcal{A}')$ and $E(\mathcal{A}'(k))$ dynamically based on the $Q(\mathcal{A})$ and $E(\mathcal{A}(k))$, when new data is sensed in grid (i_0, j_0) at time interval t_0. That is, $\mathcal{A}'$ is a copy of $\mathcal{A}$ except that $\mathcal{A}'(i_0, j_0, t_0) = \mathcal{A}(i_0, j_0, t_0) + 1$. *First*, we can directly get $Q(\mathcal{A}') = Q(\mathcal{A}) + 1$. *Second*, based on the definition of entropy: $E(\mathcal{A}(k)) = -\sum_{i,j,t} \frac{\mathcal{A}(i,j,t|k)}{Q(\mathcal{A})} \log_2 \frac{\mathcal{A}(i,j,t|k)}{Q(\mathcal{A})}$, we can derive

$$E(\mathcal{A}(k)) = \log_2 Q(\mathcal{A}) - \frac{1}{Q(\mathcal{A})} \sum_{i,j,t} \mathcal{A}(i,j,t|k) \log_2 \mathcal{A}(i,j,t|k)$$

and then $\sum_{i,j,t} \mathcal{A}(i,j,t|k) \log_2 \mathcal{A}(i,j,t|k) = Q(\mathcal{A})(\log_2 Q(\mathcal{A}) - E(\mathcal{A}(k)))$, which indicates

$$\sum_{i\neq i_k, j\neq j_k, t\neq t_k} \mathcal{A}(i,j,t|k) \log_2 \mathcal{A}(i,j,t|k) = Q(\mathcal{A})(\log_2 Q(\mathcal{A}) - E(\mathcal{A}(k))) - \mathcal{A}(i_k,j_k,t_k|k) \log_2 \mathcal{A}(i_k,j_k,t_k|k),$$

where (i_k, j_k) denotes the (i_0, j_0)'s corresponding location at granularity k; t_k refers to the t_0's corresponding time interval at granularity k. Now, for $E(\mathcal{A}'(k))$, similarly we have

$$E(\mathcal{A}'(k)) = \log_2(Q(\mathcal{A}) + 1) - \frac{1}{Q(\mathcal{A})+1} \sum_{i\neq i_k, j\neq j_k, t\neq t_k} \mathcal{A}(i,j,t|k) \log_2 \mathcal{A}(i,j,t|k) - \frac{1}{Q(\mathcal{A})+1} (\mathcal{A}(i_k,j_k,t_k|k) + 1) \log_2(\mathcal{A}(i_k,j_k,t_k|k) + 1).$$

Based on the previous formulation, we finally get

$$E(\mathcal{A}'(k)) = \log_2(Q(\mathcal{A}) + 1) - \frac{1}{Q(\mathcal{A})+1} \{Q(\mathcal{A})(\log_2 Q(\mathcal{A}) - E(\mathcal{A}(k))) - \mathcal{A}(i_k,j_k,t_k|k) \log_2 \mathcal{A}(i_k,j_k,t_k|k)\} - \frac{1}{Q(\mathcal{A})+1} (\mathcal{A}(i_k,j_k,t_k|k) + 1) \log_2(\mathcal{A}(i_k,j_k,t_k|k) + 1).$$

That is, after new data (i_0, j_0, t_0) is collected, we do find an efficient way to compute the $Q(\mathcal{A}')$ and $E(\mathcal{A}'(k))$ (i.e., the $\phi(\mathcal{A}')$) based only on current $Q(\mathcal{A})$ and $E(\mathcal{A}(k))$. Initially, we set $Q(\mathcal{A}) = 0$ and $E(\mathcal{A}(k)) = 0$, as $\mathcal{A}$ is null.

REFERENCES

1. Asaad Ahmed, Keiichi Yasumoto, Yukiko Yamauchi, and Minoru Ito. 2011. Distance and time based node selection for probabilistic coverage in People-Centric Sensing. In *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2011*. 134–142.

2. Ning Chen, Nick Gravin, and Pinyan Lu. 2011. On the Approximability of Budget Feasible Mechanisms. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, 2011*. 685–699.

3. Yohan Chon, Nicholas D. Lane, Yunjong Kim, Feng Zhao, and Hojung Cha. 2013. Understanding the coverage and scalability of place-centric crowdsensing. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2013*. 3–12.

4. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms (3. ed.)*. MIT Press.

5. Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. 2015. CityMomentum: an online approach for crowd behavior prediction at a citywide level. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2015*. 559–569.

6. Raghu K. Ganti, Fan Ye, and Hui Lei. 2011. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine* 49, 11 (2011), 32–39.

7. Sara Hachem, Animesh Pathak, and Valérie Issarny. 2013. Probabilistic registration for large-scale mobile participatory sensing. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications, 2013*. 132–140.

8. Kai Han, Chi Zhang, Jun Luo, Menglan Hu, and Bharadwaj Veeravalli. 2016. Truthful Scheduling Mechanisms for Powering Mobile Crowdsensing. *IEEE Trans. Comput.* 65, 1 (2016), 294–307.

9. Ryoma Kawajiri, Masamichi Shimosaka, and Hisashi Kahima. 2014. Steered crowdsensing: incentive design towards quality-oriented place-centric crowdsensing. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014*. 691–701.

10. Juong-Sik Lee and Baik Hoh. 2010. Dynamic pricing incentive for participatory sensing. *Pervasive and Mobile Computing* 6, 6 (2010), 693–708.

11. Evangelos Niforatos, Pedro Campos, Athanasios Vourvopoulos, André Dória, and Marc Langheinrich. 2014. Atmos: a hybrid crowdsourcing approach to weather estimation. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014*. 135–138.

12. Wentao Robin Ouyang, Animesh Srivastava, Prithvi Prabahar, Romit Roy Choudhury, Merideth Addicott, and F. Joseph McClernon. 2013. If you see something, swipe towards it: crowdsourced event localization using smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2013*. 23–32.

13. Kiran K. Rachuri, Christos Efstratiou, Ilias Leontiadis, Cecilia Mascolo, and Peter J. Rentfrow. 2013. METIS: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications, 2013*. 85–93.

14. Sasank Reddy, Deborah Estrin, and Mani B. Srivastava. 2010. Recruitment Framework for Participatory Sensing Data Collections. In *Proceedings of the 8th International Conference on Pervasive Computing, 2010*. 138–155.

15. Jingbo Shang, Yu Zheng, Wenzhu Tong, Eric Chang, and Yong Yu. 2014. Inferring gas consumption and pollution emission of vehicles throughout a city. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014*. 1027–1036.

16. Yaron Singer. 2010. Budget Feasible Mechanisms. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, 2010*. 765–774.

17. Leye Wang, Daqing Zhang, Animesh Pathak, Chao Chen, Haoyi Xiong, Dingqi Yang, and Yasha Wang. 2015b. CCS-TA: quality-guaranteed online task allocation in compressive crowdsensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2015*. 683–694.

18. Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Wasim Sadiq, and Xiaofang Zhou. 2015a. Geo-SAGE: A Geographical Sparse Additive Generative Model for Spatial Item Recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015*. 1255–1264.

19. Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014*. 25–34.

20. Daqing Zhang, Haoyi Xiong, Leye Wang, and Guanling Chen. 2014. CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014*. 703–714.

21. Fuzheng Zhang, David Wilkie, Yu Zheng, and Xing Xie. 2013. Sensing the pulse of urban refueling behavior. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2013*. 13–22.

22. Yin Zhang, Matthew Roughan, Walter Willinger, and Lili Qiu. 2009. Spatio-temporal compressive sensing and internet traffic matrices. In *Proceedings of the ACM SIGCOMM 2009 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2009*. 267–278.

23. Yu Zheng. 2015. Trajectory Data Mining: An Overview. *ACM Transactions on Intelligent Systems and Technology* 6, 3 (2015), 29.

24. Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014a. Urban Computing: Concepts, Methodologies, and Applications. *ACM Transactions on Intelligent Systems and Technology* 5, 3 (2014), 38:1–38:55.

25. Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, Yanchi Liu, and Eric Chang. 2014b. Diagnosing New York city's noises with ubiquitous data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014*. 715–725.

26. Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. 2011. Urban computing with taxicabs. In *Proceedings of the 2011 ACM Conference on Ubiquitous Computing, 2011*. 89–98.

27. Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. 2015. Forecasting Fine-Grained Air Quality Based on Big Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015*. 2267–2276.