A Privacy-Preserving Task Recommendation Framework for Mobile Crowdsourcing

Yanmin Gong*, Yuanxiong Guo†, and Yuguang Fang*
* Dept. of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA
† School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA
E-mail: ymgong@ufl.edu, richard.guo@okstate.edu, fang@ece.ufl.edu

***Abstract*—Mobile crowdsourcing enables mobile workers to complete a broad range of crowdsourcing tasks anywhere at any time. However, recommending suitable crowdsourcing tasks to mobile workers requires sensitive information such as location and activity, which raises serious privacy concerns. In this paper, we formulate the task recommendation process as an optimization problem which balances privacy, utility, and efficiency. We show that this optimization problem is NP-hard, and present a greedy solution which approximates the optimal solution within a factor of $1 - 1/e$. We also design an efficient aggregation protocol to compute statistics of mobile workers required in the optimization problem while providing strong privacy guarantee. Both numerical evaluations and performance analysis are carried out to show the effectiveness and efficiency of the proposed framework. To the best of our knowledge, our work is the first to consider privacy issues in task recommendation for mobile crowdsourcing.**

***Index Terms*—Mobile crowdsourcing, privacy, security, task recommendation**

I. Introduction

Crowdsourcing is a distributed problem-solving model in which a crowd of undefined size is engaged to solve complex problems through an open call [1]. Traditional crowdsourcing platforms are online websites such as Google MapMaker and Amazon MTurk. With the rise of the mobile era, mobile crowdsourcing has great potential to thrive. Nowadays, many people carry smartphones or other mobile devices with them wherever they go, while having no computers at home. The ubiquity and advanced sensing capabilities of mobile devices enable users to share richer information anywhere at any time, and therefore complete a broad range of tasks. Attracted by these opportunities, several mobile crowdsourcing applications have emerged on topics such as traffic monitoring (e.g., VTrack [2]) and indoor localization (e.g., Airplace [3]).

In crowdsourcing applications, users who accomplish tasks can get paid for qualified work. Since there are huge amount of available tasks (e.g., more than $280,000$ tasks for MTurk), it is cumbersome to find the "right" task to accomplish. On traditional crowdsourcing platforms, workers may search for tasks by themselves. However, for mobile users, such task search process can be quite inefficient due to the limited screen and keyboard on a mobile phone. Moreover, many mobile crowdsourcing applications are time-sensitive, which means that tasks should be actively pushed to the crowd in order to receive timely responses. These factors underscore the need for task recommendation service by mobile crowdsourcing platforms.

Task recommendation systems actively recommend tasks based on the contexts of workers such as location and activity. For example, the tasks to monitor noise at night are only recommended to residents in the target area. These contexts contain sensitive information that can be used to uniquely identify an individual. As a result, workers may be reluctant to share such information. Motivated by this observation, we propose a privacy-aware task recommendation framework.

The proposed task recommendation framework (1) models how to select tasks based on the context of a worker, and (2) gathers statistics of user contexts used in the previous model. Intuitively, both modules require access to private information. However, in our framework, workers can decide how much information they would like to share with the platform. In the task selection module, a worker can control how detailed his context is revealed to the server, and receive recommendations based on the information he provides. We show the trade-off among privacy, utility and efficiency in this model, where privacy indicates how much information is shared, utility indicates the usefulness of the recommendations, and efficiency refers to the number of tasks recommended at a time. We jointly consider the three aspects and formulate the task selection process as an optimization problem that maximizes the total expected utility of the recommended tasks with constraints on privacy and efficiency. For statistics collection, workers can choose whether to contribute his data or not, and differential privacy is guaranteed if they choose to do so.

Related Works. There are a few works in task recommendation for web-based crowdsourcing applications [4], [5]. These works assign heterogeneous tasks to workers based on their skill sets and interests. For mobile crowdsourcing, however, task recommendation should be based on sensitive information such as location and activity, which has not been addressed by these works. Previous works on privacy issues in mobile applications mainly focus on location privacy [6], [7] and preserve privacy through obfuscation or aggregation approaches. None of these works discuss how to recommend tasks in the absence of accurate information.

The remainder of this paper is organized as follows. We first present our framework in Section II. Then we describe the task selection module as a constrained optimization problem in Section III. Section IV develops an approximation algorithm to solve the optimization problem. Section V presents a privacy-preserving protocol for statistics gathering, and experimental

This work was supported in part by the U.S. National Science Foundation under grants CNS-1343356 and ECCS-1129062.

978-1-4799-3512-3/14/$31.00 ©2014 IEEE

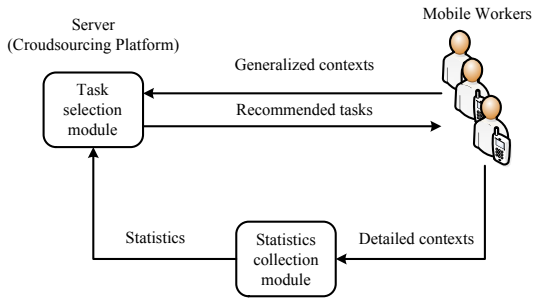

Fig. 1. Framework of the recommendation system

results and performance analysis are given in Section VI. Section VII concludes the work.

II. Framework

A. Overview

There are three parties in a typical mobile crowdsourcing setting: task requesters, mobile workers (hereafter also referred to as users), and the crowdsourcing platform. The task requester initiates crowdsourcing tasks by submitting them to the crowdsourcing platform. The crowdsourcing platform then recommends a list of suitable tasks for each mobile worker, pushing the list to his mobile device. When the mobile worker receives a list of recommended tasks, he chooses a task, completes it within deadline, and returns the answer to the platform. The platform forwards the answer to the task requester, who decides whether or not the answer is qualified. The user receives payments for qualified answers, and the platform also receives a portion of the payments as its commission. The focus of this paper is the task recommendation part, where tasks are selected and pushed to a user based on his context.

B. Privacy Model

As shown in Fig. 1, the proposed task recommendation framework consists of two modules, i.e., task selection and statistics collection. In the first one, the server tailors a list of recommended tasks for a specific user based on his context. The user context may contain sensitive information which may uniquely identify a user, such as location and activity. Hence, users may be reluctant to share this information with the server. Instead of providing a detailed context, we allow the user to share only a generalized context with the server. The user is able to decide which level of generalization he is comfortable with based on the sensitivity of his detailed context. For example, if he is walking in a crowded city, he may be comfortable to share a detailed location without the concern of being identified; on the other hand, if he is passing through a rural area, he may want to provide a generalized location to protect his privacy.

The statistics collection module gathers statistics that are used in the first module. Statistics are gathered by asking users to vote based on their detailed contexts and historical performance. This process is optional for a user, so a user can decide whether or not to contribute his data. But in either way, his privacy is guaranteed. The privacy guarantee provided for participating users is (ϵ, δ)-differential privacy [8], which discourages adversaries with arbitrary background knowledge. Formally speaking,

***Definition** 1:* A statistics collection algorithm $\mathcal{F}$ satisfies (ϵ, δ)-differential privacy if for any two datasets D_1 and D_2 which differ in one row, the following inequality holds:

$$\Pr[\mathcal{F}(D_1) \in O] \leq \exp(\epsilon) \times \Pr[\mathcal{F}(D_2) \in O] + \delta, \quad (1)$$

where $O \subseteq \mathrm{range}(\mathcal{F})$.

In the definition, ϵ bounds the ratio of probability distributions of two datasets differing in at most one row, and δ permits us to relax the relative shift at events that are not likely to happen, bounding the probability of a privacy breach. If the outputs of statistics collection module achieve (ϵ, δ)-differential privacy, the fact whether a user provides information or not to the recommendation server will not change the server's knowledge on him.

III. Optimization Model for Task Selection

A. Definitions

Before proceeding further, we give a list of definitions used in the paper as follows.

***Definition** 2:* Contexts and Tasks

- Denote by $\mathcal{C} = \{c : c = 1, 2, \ldots, C\}$ the set of all detailed contexts. Each worker has a detailed context c.
- Denote by $\hat{\mathcal{C}} = \{\hat{c} : \hat{c} = 1, 2, \ldots, \hat{C}\}$ the set of all generalized contexts. Each detailed context is transformed into a generalized context, and a generalized context corresponds to multiple detailed contexts.
- Denote by $\mathcal{T} = \{t : t = 1, 2, \ldots, T\}$ the set of all tasks. For simplicity of notations, we treat tasks that have the same requirements for user contexts and the same payment as one task. Each task may have multiple instances. The payment for a task t is denoted as ρ_t.

***Definition** 3:* Click and Approval Rate (CAR): The platform can earn the commission for a task only when the task has been completed and the answer has been approved. These two conditions are characterized by the click through rate and the approval rate, respectively. We define a metric CAR that combines these two rates as follows.

- CAR is calculated as W_1, the total number of users with context c who have completed task t, divided by W_2, the total number of all users with context c, i.e., $\mathrm{CAR}(t|c) = W_1/W_2$.

B. Trade-Off

The optimization model of task selection specifies how to choose tasks based on limited information about a user. There are three conflicting design goals in this model: utility, privacy, and efficiency.

Utility. Utility represents the usefulness of a list of recommended tasks, which is reflected in the revenue obtained by recommending the tasks. For the crowdsoucing platform, higher utility means larger commission. For the user, higher utility means better payment. Note that only when a task is completed and the result is approved will the platform and the user get paid.

Privacy. User contexts used for task recommendation may contain sensitive information that leads to identification of a

specific user. To reduce the risk of being identified, instead of providing the detailed contexts, a user provides only a generalized context which obfuscates privacy sensitive information such as location and activity. A more generalized context leads to a higher level of privacy.

Efficiency. The number of recommended tasks directly influences the easiness for users to select a task from the list of recommended tasks. Larger number of tasks needs more time to read and select, thus leading to lower efficiency for task selection at the user side. Therefore, the recommendation system should recommend a small amount of tasks at a time for efficiency concern.

It can be easily shown that these three goals cannot be optimized simultaneously. Consider the case when utility and efficiency are maximized, then the server recommends only one task with a maximized utility for the user. However, this can be hardly achieved if the server does not have a detailed knowledge on user context. Hence, a maximized utility and efficiency leads to a low level of privacy. To jointly consider these three aspects, we formulate the task selection process as an optimization problem that maximizes the total expected utility of the recommended tasks with constraints on privacy and efficiency.

C. Optimization Model

In the task selection process, a user first shares some information on his contexts with the server. The server then selects L tasks based on this information, and sends them to the user. Lastly, the user selects a task from the recommended list, and completes it. Therefore, the final task is selected jointly by the server and the user.

As mentioned before, there are three conflicting goals. Although these goals cannot be optimized simultaneously, there are several candidate objective functions, which optimizes the goals from different aspects. In the following, we choose an optimization objective function that represents the utility, and model the other two goals (i.e., efficiency and privacy) as constraints.

In the optimization model of recommending tasks, the server needs to select L tasks that maximize the utility of the crowdsourcing platform based on the partial information $\hat{c}$ given by the user. We assume that the server has prior knowledge on the statistics, i.e., CARs and distributions of detailed contexts. For the crowdsroucing platform, the utility is the expected commission of the list of tasks. Since the crowdsourcing platform gets commission when the user selects a task and returns a qualified answer, the amount of commission is determined by how the user behaves. Assume that a user selects a task that maximizes his own revenue based on his detailed context c. Consider the probability of all detailed contexts that generalize into $\hat{c}$, the expected commission of a list of recommended tasks T is defined as follows:

$$\mathbb{E}[\text{Commission}(T|\hat{c})] = \sum_{c:c\to\hat{c}} \Pr[c|\hat{c}] \cdot \alpha \cdot \max_{t\in T} \rho_t \cdot \text{CAR}(t|c), \tag{2}$$

where α is the portion of revenue that the platform can obtain for each successful transaction. As we have discussed before, L reflects the efficiency of task selection at the user side, and $\hat{c}$ is chosen by the user according to his privacy concern. Given L and $\hat{c}$, the server needs to select a set of tasks which maximize the above objective function, i.e.,

$$T^* = \operatorname*{argmax}_{T\subseteq\mathcal{T}:|T|=L} \mathbb{E}[\text{Commission}(T|\hat{c})]. \tag{3}$$

When a user receives a list of recommended tasks, he selects the task with the maximized utility for him, which can be modeled as

$$t^* = \operatorname*{argmax}_{t\in T} \rho_t \cdot \text{CAR}(t|c). \tag{4}$$

IV. Algorithm for the Optimization Problem

In this section, we examine the hardness of the proposed optimization model and present a greedy approach to solve the model efficiently. The optimization problem at the user side can be easily solved because he only needs to choose a task among L tasks, where L is usually small. On the other hand, it is nontrivial for the server to select L tasks from the entire task space $\mathcal{T}$. Actually, we have the following fact:

***Proposition** 1:* Given a generalized context $\hat{c}$, it is NP-hard to find a set of tasks T^* such that:

$$T^* = \operatorname*{argmax}_{T\subseteq\mathcal{T}:|T|=L} \sum_{c:c\to\hat{c}} \Pr[c|\hat{c}] \cdot \alpha \cdot \max_{t\in\mathcal{T}} \rho_t \cdot \text{CAR}(t|c). \tag{5}$$

Proof: The proposition can be deduced from the hardness of the maximum coverage problem. Given a collection of sets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$ over some finite universe $\mathcal{U}$ and a number L as input, the maximum coverage problem wants to select at most L of these sets such that the maximum number of elements in the universe $\mathcal{U}$ are covered.

Consider the settings of task selection problem as follows: the set of contexts that generalize to $\hat{c}$ equals the universe $\mathcal{U}$. For each context $c \in \mathcal{U}$, we set $\Pr[c|\hat{c}] = 1/|\mathcal{U}|$. For each set s in the collection $\mathcal{S}$, we define a task t such that for all elements $c \in s$, $\text{CAR}(t|c) = 1$ and 0 for all other elements. Moreover, we set $\rho_t = 1, \forall t$ and $\alpha = 1$. Then, for a given set T of L tasks, the expected commission is represented as:

$$\sum_{c:c\to\hat{c}} \frac{1}{|\mathcal{U}|} \max_{t\in T} \text{CAR}(t|c), \tag{6}$$

which can be viewed as the total number of elements covered by the corresponding set of sets S divided by the universe size. By using the above transformation, any instance of the maximum coverage problem can be reduced to a instance of our task set selection problem in polynomial time. Therefore, the task recommendation problem is harder than the maximum coverage problem. Since the maximum coverage problem is known to be NP-hard, our problem is also NP-hard. ■

To solve the NP-hard problem (5) efficiently, we design a greedy heuristic algorithm as shown in Algorithm 1 below.

By repeatedly choosing a task that maximizes the utility improvement, the greedy algorithm can be proved to approximate the optimal value within $1 - 1/e$. Due to space limit, the proof of this approximation ratio is omitted here for brevity.

Algorithm 1: Greedy Algorithm of Task Set Selection for Maximizing Profit

Input: all available task set $\mathcal{T}$, generalized context $\hat{c}$, and set cardinality L
Output: recommended task set T
Initialize $T = \emptyset$;
Define $F(T) \doteq \mathbb{E}[\text{Commission}(T|\hat{c})]$;
repeat
  Choose $t \in \mathcal{T}$ that maximizes the marginal utility $F(T \cup t) - F(T)$;
  Let $T \leftarrow T \cup \{t\}$;
until $|T| = L$;
return T;

V. Privacy-Preserving Statistics Aggregation

Note that in previous sections, we assume that the server has prior knowledge on statistics such as $\Pr[c|\hat{c}]$ and $\text{CAR}(t|c)$. These statistics can be computed by counting the number of users based on their detailed contexts and CARs. In this section, we describe how to collect these statistics while preserving user privacy.

Design Goals. There are mainly three design goals for statistics collection: Firstly, user privacy should be protected during statistics collection. Secondly, the final statistics should not be distorted by a small portion of malicious users. Thirdly, the system should be scalable to a large population of users, which means that the proposed protocols should be highly efficient.

The statistics collection module consists of three parties, *users*, *server*, and *proxy*. Users contribute data regarding their detailed contexts and CARs. The server collects the answers and transforms them into statistics. The proxy is a new party who mediates between users and the server, and adds noise to the statistics. The proxy is introduced to achieve differential privacy under a distributed setting at an acceptable cost.

A. Adversarial Model

We assume that the server is malicious and seeks to compromise user privacy with or without the help of dishonest users. Users are potentially malicious. They may generate false or illegitimate data to distort the final statistics. We also introduce an "honest-but-curious" proxy to help aggregate user data and preserve differential privacy in the distributed setting. The proxy may be paid by the server but acts as an independent party, which means it will not collude with the server.

B. A Privacy-Preserving Aggregation Protocol

To calculate a statistic, the server sends queries which describes a certain status to users. Users would return "yes" or "no" answers. The statistic $\Pr[c|\hat{c}]$ is computed as the ratio of the number of users with context c to the number of users with generalized context $\hat{c}$. The server sends queries "Are you in context c?" to all the users whose generalized context is $\hat{c}$. Users respond with either 1 ("yes") or 0 ("no"). Then the server can get $\Pr[c|\hat{c}]$ by dividing the number of "yes" by the number of total answers. The other statistic $\text{CAR}(t|c)$ is calculated as the ratio of the number of completed tasks among users with context c to the total number of users with context c. Similarly, the server first sends a query which contains the following three "yes" or "no" questions, representing three possible statuses of the user: (1) The context of the user is c, but the user does not complete task t; (2) The context of the user is c, and the user completes task t; (3) The context of the user is not c. When a user receives such a query, he responds with a vector of 3 bits, each indicating the answer to one of the three questions.

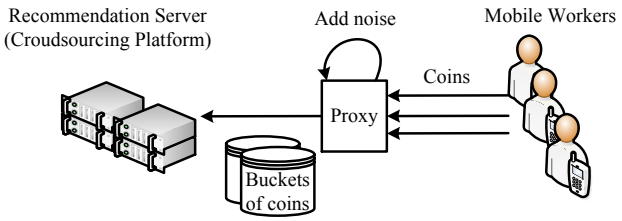

Fig. 2. The framework of the aggregation process

Now, we describe the detailed protocols of the statistics gathering process. When the server needs to calculate a statistic, it constructs a query request with a sequence number SN. The query is forwarded to the proxy. The proxy would broadcast the query to target users whose contexts are related to the query, only when the number of target users is large enough. If a user decides to respond with an answer, he constructs a vector which includes his answers. Before sending the vector out, the user encrypts the vector bit by bit with the public key of the server and attaches the sequence number SN to the ciphertext. We call the resulting message a coin. For example, if the vector is $(0, 0, 1)$, the final coin would be $e(0)||e(0)||e(1)||SN$, where $e(\cdot)$ is the encryption operation under the public key of the server.

The coin is then encrypted once more with the public key of the proxy, and sent to the proxy. The proxy decrypts all the messages, groups the coins with the same sequence number, puts them in a bucket, and sends them to the server. The proxy does not know the bits encrypted in the coin, hence it gains no additional information on the detailed contexts of users. After the server gets enough coins from the proxy, it decrypts all the coins and calculates the votes for a specific request. Once calculated, the statistics can be updated at low cost, considering that only newly collected coins need to be decrypted. Since the server cannot tell who constructs the coins, the identities of users are anonymized. The aggregation framework is illustrated in Fig. 2.

C. Generating Encrypted Answers

We propose to use the Goldwasser-Micali (GM) cryptosystem [9] for generating the coins. It is designed to encrypt data bit by bit, which fits our scenario well. An interesting property of this scheme is that it is a homomorphic encryption scheme, where the result of addition on plaintexts can be obtained by decrypting the result of computations on the ciphertexts. In the GM cryptosystem, the plaintext is either 1 or 0. Other forms of plaintext can be easily detected and discarded. Hence, a malicious user could only distort the final result by at most one.

The protocol of GM encryption is as follows.

- *Key generation*: Let $N = p \cdot q$, where p and q are two large primes independent of each other. Choose a non-residue x such that the Legendre symbol $x/p = x/q = -1$ and hence the Jacobi symbol $x/N = +1$. The public key is (N, x), and the private key is (p, q).
- *Encryption*: Let b be the message we want to encrypt. Choose a non-zero random number $r \in \mathbb{Z}_N^*$. The ciphertext e is given by

$$e = r^2 \cdot x^b \mod N. \tag{7}$$

- *Decryption*: Given the ciphertext e, the receiver uses the prime factorization (p, q) to check whether e is a quadratic residue $\mod N$. In order to do this, he first computes $e_p = e \mod p$ and $e_q = e \mod q$. If both $e_p^{(p-1)/2} \equiv 1 \mod p$ and $e_q^{(q-1)/2} \equiv 1 \mod q$ hold, e is a quadratic residue $\mod N$. He then sets $b = 0$ if the result is yes, and sets $b = 1$ otherwise.

In the GM cryptosystem, the legitimacy of the ciphertext can be checked by calculating the Jacobi symbol: only the ciphertext whose Jacobi symbol equals $+1$ is legitimate. Hence with this cryptosystem, we can easily achieve one of the design goals, i.e., the final statistics should not be distorted by a small portion of malicious users. Since a user can distort the final result by at most 1, the deviation of the final result is bounded by m if there are at most m malicious users.

D. Preserving Differential Privacy

With the GM cryptosystem, the answers from users are aggregated so that individual users are anonymized from the server. However, this approach alone cannot protect against malicious servers who have auxiliary information. Hence, we further add differentially private noise to the statistics with the help of the proxy. The amount of noise required to achieve (ϵ, δ)-differential privacy is calculated in [8], and described as follows.

***Proposition** 2:* Let n be the number of unbiased coins added in a bucket, i.e., the amount of Binomial noise. The statistics collection algorithm achieves (ϵ, δ)-differential privacy if

$$n \geq \frac{64 \ln(\frac{2}{\delta})}{\epsilon^2}. \tag{8}$$

The parameter δ is selected by the server. Let M be the total number of queried users. Suppose that every query of every person is sensitive, then $\delta > 1/M$ indicates the disclosure of at least one person's privacy. Therefore, δ is selected to be smaller than $1/M$. With this constraint, the amount of noise for each bucket should satisfy

$$n \geq \frac{64 \ln(2M)}{\epsilon^2}. \tag{9}$$

Hence, to achieve (ϵ, δ)-differential privacy, the proxy generates n noisy coins following a Binomial distribution which yields success with probability 0.5, and adds the noisy coins to each bucket. In order to generate unbiased and blinded coins, the proxy should collaborate with users. This is because if the proxy generates the unbiased coins by itself, it would know the accurate value of the noise, and then when the server publishes the obfuscated statistics, the proxy would be able to recover the accurate statistics. On the other hand, if the users are trusted to generate the noise coins alone, they may intentionally distort the final statistics by generating biased noise coins. Hence, following a flipping approach in [10], we let the users generate n coins first, which are flipped by the honest-but-curious proxy. This is possible due to the following homomorphic property of the GM cryptosystem:

Homomorphic property of GM cryptosytem. For any $b_1, b_2 \in \{0, 1\}$, we have $\mathcal{E}(b_1) \cdot \mathcal{E}(b_2) = \mathcal{E}(b_1 \oplus b_2 \mod N)$, where $\mathcal{E}(\cdot)$ is the encryption operation of the GM cryptosystem.

In the XOR operation, as long as one of the two operators is unbiased, then the final result would be unbiased. Hence multiplying a coin generated by users and a unbiased coin generated by the proxy results in a flipped coin that is both unbiased and hidden from the proxy. In this way, we can generate a pool of unbiased coins for noise addition.

When the server receives the noisy answer for a certain request with sequence number SN, it first decrypts all the coins with its secret key and sums up the plaintexts. Then it can recover the statistics unbiasedly by subtracting $n/2$ from the sum, where n is the number of noise coins. As described at the beginning of this section, both $\Pr[c|\hat{c}]$ and $\text{CAR}(t|c)$ can be calculated following the protocols.

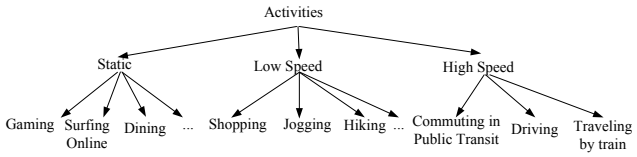

Fig. 3. A taxonomy of user activities

VI. Experiments

Due to the lack of real-world data on crowdsourcing tasks, we evaluate our proposed method based on synthetically generated data sets. The details are given in the following.

A. Experimental Setup

Context instantiations. Users can generalize their detailed contexts according to a hierarchy predefined by the recommendation system. Quantifiable contexts can be easily divided into multiple levels. For example, the location information, represented in the form of (latitude, longitude), can be divided into 5 levels. If a user chooses level-a generalization (a ranges from 0 to 4), he should keep a decimal digits for both the latitude and the longitude. If a context is not quantifiable, it can be described in a tree taxonomy. Fig. 3 shows an example, which describes activities with different precisions. If a user chooses level-b precision to describe his current or future activities, but does not want the server to know in detail (b ranges from 0 to 2), he can describe it using the descriptions in the level-b nodes. For example, if a user dining in a restaurant chooses level-2 generalization, he would only tell the system server that he is static.

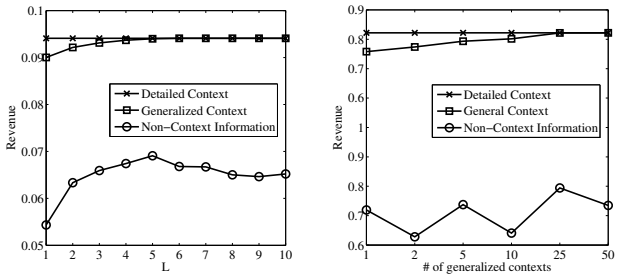

Fig. 4. (a) Impact on utility when varying the size of the list L; (b) Impact on utility when varying the total number of generalized contexts

We assume such a task selection setting: a population of users with 100 different detailed contexts are being recommended from a pool of 1000 tasks. Statistics $\Pr(c|\hat{c})$ and CAR are generated by random functions. We compare the final revenue obtained by three approaches: task selection with detailed context, task selection with generalized context (our approach), and task selection with no information on user context. For the second approach, we use Algorithm 1 to get an approximate solution.

B. Numerical Evaluation

We evaluate the performance of the above approaches in terms of the total revenue of users, which is proportional to the commission of the crowdsourcing platform. Fig. 4(a) shows the impact on the total revenue when varying the size of the list L. We can see that the revenue of the proposed approach increases as L increases, and converges to the first approach (task selection approach with detailed context) after L exceeds 4. This is reasonable because when L is small, users would have more tasks to choose from, resulting a higher utility, but when L exceeds certain threshold, the added tasks result in little marginal improvement on utility, and the proposed approach performs the same as the first approach, i.e., task selection with detailed context.

We divide detailed contexts into subsets, each corresponding to a generalized context. If the number of resulting generalized contexts is small, users are provided a high level of privacy. Fig. 4(b) shows the impact of varying privacy levels on the expected commission with L fixed at 3. We can see that when privacy level decreases, the gap between task selection with detailed context and with generalized context becomes small. However, the gap is always within a small range, which indicates that the performance of our privacy-preserving approach is close to that of the privacy-oblivious one.

C. Performance Analysis

Now we analyze the computation overheads of the proposed aggregation protocols, where a lot of encryptions and decryptions are involved. We assume the following statistics collection setting: a population of 1000 users have the same generalized context, which can be further divided into 100 different detailed contexts. For the specific generalized context, there are 90 tasks. We also assume that the key length for the GM cryptosystem is 1024-bit. Suppose 10% of the users participate in the statistics collection process, the proxy needs to execute 340 encryptions and 340 homomorphic XORs for a single statistic query when the privacy parameter ϵ is set to 1 according to (9).

A smartphone running Android 2.2 with 1GHz processor can execute 800 GM encryptions per second [10]. Hence the computation overhead for mobile users is negligible. A proxy needs to execute $340 * 27200$ encryptions and $340 * 27200$ homomorphic XORs in the assumed setting; if we implement the proxy with Apache Tomcat 6.0.33, it takes 10 minutes and 75 seconds, respectively. The server needs to carry out $840 * 27200$ decryptions; if it is implemented with Apache Tomcat 6.0.33, it takes 58 minutes to complete all the executions. Note that the statistics only needs to be calculated once. After it has been calculated, it can be updated at a low frequency, which involves even lower overheads.

VII. Conclusion

We have addressed in this paper the privacy issues in task recommendation for mobile crowdsourcing. We have proposed a framework that describes the trade-off among utility, privacy, and efficiency. We have shown that the task set selection problem under privacy and efficiency constraints is NP-hard and then proposed an approximation algorithm to solve it. We have also proposed a privacy-preserving protocol for statistics collection in our framework. Both numerical experiments and performance analysis are conducted to demonstrate the effectiveness of our proposed approach.

References

[1] G. Chatzimilioudis, A. Konstantinidis, C. Laoudias, and D. Zeinalipour-Yazti, "Crowdsourcing with smartphones," *Internet Computing, IEEE*, vol. 16, no. 5, pp. 36–44, 2012.

[2] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2009, pp. 85–98.

[3] C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti, and C. G. Panayiotou, "The airplace indoor positioning platform for android smartphones," in *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*. IEEE, 2012, pp. 312–315.

[4] M.-C. Yuen, I. King, and K.-S. Leung, "Task recommendation in crowdsourcing systems," in *Proceedings of the First International Workshop on Crowdsourcing and Data Mining*. ACM, 2012, pp. 22–26.

[5] V. Ambati, S. Vogel, and J. Carbonell, "Towards task recommendation in micro-task markets," in *Proceedings of The 25th AAAI Workshop in Human Computation*, 2011.

[6] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Pervasive computing*. Springer, 2005, pp. 152–170.

[7] J. W. Brown, O. Ohrimenko, and R. Tamassia, "Haze: Privacy-preserving real-time traffic statistics," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 530–533.

[8] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology-EUROCRYPT 2006*. Springer, 2006, pp. 486–503.

[9] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. ACM, 1982, pp. 365–377.

[10] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke, "Towards statistical queries over distributed private user data," in *Proceedings of the 9th Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.