Expertise-Aware Truth Analysis and Task Allocation in Mobile Crowdsourcing

Xiaomei Zhang*, Yibo Wu†, Lifu Huang‡, Heng Ji‡ and Guohong Cao†
*Department of Computer Science, University of South Carolina Beaufort
†Department of Computer Science and Engineering, The Pennsylvania State University
‡Computer Science Department, Rensselaer Polytechnic Institute
∗xiaomei@uscb.edu, †{ywx185,gcao}@cse.psu.edu, ‡{huangl7,jih}@rpi.edu

Abstract—In mobile crowdsourcing, the accuracy of the collected data is usually hard to ensure. Researchers have proposed techniques to identify truth from noisy data by inferring and utilizing the reliability of mobile users, and allocate tasks to users with higher reliability. However, they neglect the fact that a user may only have expertise on some problems (in some domains), but not others, and hence causing two problems: low estimation accuracy in truth analysis and ineffective task allocation. To address these problems, we propose Expertise-aware Truth Analysis and Task Allocation (ETA²), which can effectively infer user expertise, and then estimate truth and allocate tasks based on the inferred expertise. ETA² relies on a novel semantic analysis method to identify the expertise, and an expertise-aware truth analysis method to find the truth. For expertise-aware task allocation in ETA², we formalize and solve two problems based on the optimization objectives: max-quality task allocation which maximizes the probability for tasks to be allocated to users with high expertise and min-cost task allocation which minimizes the cost of task allocation while ensuring high-quality data are collected. Experimental results based on two real-world datasets and one synthetic dataset demonstrate that ETA² significantly outperforms existing solutions.

1 INTRODUCTION

Today’s mobile devices, such as smartphones, smart watches and tablets, possess excellent capabilities in sensing and communication. Thanks to the prevalence of these mobile devices, recent years have seen the emergence of many mobile crowdsourcing applications [1][2][3]. With mobile crowdsourcing, people are able to collect and share data for various tasks, such as reporting the latest price of some specific product in supermarkets. In some other apps, such as map based service, through crowdsourcing, real-time traffic conditions are monitored and shared with drivers.

The quality of crowdsourcing depends on the accuracy of the collected data. However, in the real world, data accuracy is hard to ensure due to many reasons, such as the limited sensing capability, the unreliable data source, and some uncontrollable subjective factors. For example, accurate noise level may not be obtained if the reporting user does not have the right sensor installed in the mobile device. A user may intentionally generate data instead of performing the task at the specified location to save time, effort, or resources such as battery. To address this problem, it is crucial to estimate the truth (i.e., the accurate data) based on the collected data by applying appropriate truth analysis techniques. In addition to truth analysis, it is equally important to allocate tasks to appropriate users so that high-quality data can be collected.

Existing techniques on truth analysis and task allocation focus on studying the reliability of users, based on the assumption that high-reliability users tend to provide high-quality data. By inferring and utilizing user reliability, the truth can be better identified by assigning higher weights to users with higher reliability [4][5][6]. Similarly, by considering user reliability, tasks can be allocated to users with higher reliability, in order to collect high-quality data [7][8][9].

However, these existing techniques are based on the assumption that the reliability of a user does not change with tasks, while neglecting the fact that a user may only have expertise on some problems (in some domains), but not others. Neglecting this expertise diversity may cause two problems: low estimation accuracy in truth analysis and ineffective task allocation. This is because a user inferred to have high reliability may actually have low expertise in some domains, and provide low-quality data for tasks in these domains. A more severe problem is related to unfair task allocation, i.e., all tasks are assigned to a few users with “high-reliability” while the remaining majority of users are not assigned with any tasks.

Considering these problems, it is important to design expertise-aware solutions for truth analysis and task allocation. However, it is a challenge to design expertise-aware solutions due to the following three reasons. First, it is hard to identify user expertise and the expertise domains of the tasks without any prior knowledge on user behaviors and the ground truth of the tasks. Second, allocating tasks to users with the highest expertise is hard to achieve in many cases. This is because some users may have high expertise in multiple domains, but only have limited processing capability, and therefore can not finish all the assigned tasks
within the time limit.

Besides these challenges related to expertise, we have to address the challenge of the task allocation cost generated at the server. Task allocation involves user recruiting, which usually incurs non-negligible cost on the server side. For example, each user may be paid a specific amount if the user is recruited or finishes the task. Neglecting the cost may result in more users being assigned to tasks, generating unnecessary cost to the server. Thus, it is a challenge to minimize the task allocation cost while ensuring high-quality data are collected.

In this paper, we propose the Expertise-aware Truth Analysis and Task Allocation (ETA²) approach to address the aforementioned challenges. ETA² can effectively infer user expertise and then allocate tasks and estimate truth based on the inferred expertise. If task allocation involves cost, ETA² can also effectively reduce the cost of task allocation. More specifically, we have the following contributions:

- We first propose techniques to identify the expertise domains of the tasks. Specifically, we design a novel semantic analysis method to extract and quantify the semantic information of tasks based on the task descriptions. Then, we propose a dynamic hierarchical clustering approach to cluster tasks based on their extracted semantic information, so that each cluster corresponds to one expertise domain and the tasks inside the cluster belong to the corresponding expertise domain.

- For expertise-aware truth analysis, we build expertise-aware statistical models by combining the expertise models of users and tasks, and apply maximum likelihood estimation (MLE) to estimate the truth and learn user expertise.

- For expertise-aware task allocation, we formalize and solve two problems. The first problem is max-quality task allocation, which maximizes the probability for tasks to be allocated to users with high expertise while ensuring the workload does not exceed the processing capability of each user. This problem improves data quality, but neglects the high-cost of task allocation (e.g., payment to the users). To consider such cost, we study the second problem of min-cost task allocation which minimizes the cost of task allocation while ensuring high-quality data are collected.

The rest of the paper is organized as follows. Section 2 introduces some background and gives an overview of ETA². Section 3 presents how to identify the expertise domains of the tasks. Section 4 presents expertise-aware truth analysis. Section 5 formalizes and solves the two problems of expertise-aware task allocation. Evaluation results are presented in Section 6. Section 7 reviews the related work and Section 8 concludes the paper.

2 PRELIMINARY

2.1 Background

The system we consider includes a crowdsourcing server and multiple users who are able to communicate with the server using their mobile devices. A group of tasks is created at the server at each time step (e.g., each day). After a task is created, the server allocates it to selected users specifying the required data, the task deadline and the estimated processing time required for completing the task. Multiple users may be queried for each task considering that the data collected from a single user may be inaccurate. Then, users collect data as specified by the task descriptions and send data back to the server. After receiving all provided data for a task, the server estimates the truth using a truth-estimation technique.

Without loss of generality, we assume the processing capability is limited at each user, i.e., only limited time can be used for collecting data during each time step. Therefore, only limited number of tasks can be performed at each user.

In this paper, we consider the collected sensing data to be numerical values. Then, the magnitude of the data may vary tremendously for different tasks. To ensure different tasks can be processed using the same statistical model, the data value for a task is normalized.

2.2 An Overview of ETA²

Figure 1 shows an overview of ETA², which has the following three main modules:

- Identifying Task Expertise (Module 1): This module is used to find the expertise domains of the tasks.

- Expertise-aware Truth Analysis (Module 2): This module is used to infer the truth after data for the tasks have been collected from users. In the process of truth analysis, the expertise of users can also be identified.

- Expertise-aware Task Allocation (Module 3): This module is used to assign the newly created tasks to users according to their expertise. According to the objective, different problems are formalized, including max-quality task allocation, and min-cost task allocation, as specified in Section 5.

The process as shown in Figure 1 starts with a warm-up period, after the Start button. When the system first starts, there is no prior knowledge about user expertise. Thus, in the warm-up period, the tasks are allocated to users randomly. After data have been collected from users, user expertise can be learned. The learned user expertise can be further utilized for task allocation.

After the warm-up period, ETA² starts a repetitive process, as shown inside the dotted square in Figure 1. When new tasks are created in each time step, the server first finds the expertise of each task. Then, the server applies the expertise-aware task allocation technique to assign tasks to users with the right expertise. Then, users collect data as specified by the task description and send the data back to the server. After receiving the data, the server estimates the truth by applying expertise-aware truth analysis techniques and updates user expertise by incorporating the newly collected data.

In the rest of the paper, the three modules of Figure 1 are explained. More specifically, Section 3 focuses on identifying task expertise. Section 4 focuses on expertise-aware truth analysis. Section 5 focuses on expertise-aware task allocation. Section 5.1 and Section 5.2 present max-quality task allocation and min-cost task allocation, respectively.
2.3 Distribution of Random Observation

In this paper, we assume the random observations of users for a task follow normal distribution. To validate this assumption, we conduct an experiment to find the distribution of the observation error based on two real-world datasets (survey-based dataset and SFV dataset). For better presentation, more detailed information about the two datasets is not presented here, but can be found in Section 6. The observation error of a user $i$ for a task $j$ is simply computed as the difference between the observation $x_{ij}$ and the ground truth $\mu_j$ of the task divided by the standard deviation $std_j$ among all observations for task $j$:

$$err_{ij} = \frac{x_{ij} - \mu_j}{std_j}.$$  

We then get the distribution of observation errors by accumulating users’ observations for all tasks. Figure 2 shows the result. The figure also shows the probability density function of the standard normal distribution. As can be seen, the error of the observed data follows the standard normal distribution very well. This result implies that the random observations of users can be approximated by normal distribution very well. This result implies that the random observations for all tasks follow normal distribution. The figure also shows the probability density function of the standard normal distribution. As can be seen, the result. The figure also shows the probability density function of the standard normal distribution. As can be seen, the result.

![Error distribution in datasets](image)

**Fig. 2: The observation error follows normal distribution.**

2.4 Expertise Model

In our expertise model, there are $\mathcal{D}$ expertise domains, where $\mathcal{D}$ is not fixed and may be increased when new tasks are added to the system. Each sensing task $j$ belongs to one expertise domain, denoted as $d_j$. The expertise profile of a mobile user $i$ is represented by a $\mathcal{D} \times 1$ vector:

$$U^i = [u_{i1}, u_{i2}, ..., u_{i\mathcal{D}}]^T \quad (1)$$

where $u_{ik}$ is user $i$’s expertise in domain $k$ ($u_{ik} \geq 0$). A higher $u_{ik}$ means more expertise. $u_{ik} = 0$ means $i$ has no expertise in domain $k$. A user may have expertise in multiple domains.

The expertise of user $i$ for task $j$, represented as $u_{ij}$, determines the quality of the data provided by the user. Having higher expertise means that the user is likely to provide higher-quality data for the task. A user with lower expertise is likely to provide data deviating from the ground truth. Specifically, we assume that the observation of user $i$ for task $j$ follows normal distribution $N(\mu_j, (\sigma_j/u_{ij})^2)$, where $\mu_j$ is the ground truth of task $j$ and $\sigma_j/u_{ij}$ is the standard deviation. Here, $\sigma_j$ is the base number of task $j$, which is used to normalize the data value of task $j$. $\sigma_j$ is unknown but can be learned as presented in Section 4.

3 Task Expertise Identification

3.1 Basic Idea

To design expertise-aware truth analysis, we need to first identify the expertise domains of the tasks. Since there is no existing expertise information, we cannot simply classify tasks to existing expertise domains. The only information available for a task is the task description. There are many observations for each individual task still follows normal distribution. Chi-square goodness-of-fit test is applied to test the normality. In the test, the null hypothesis is that the observations for a task comes from a normal distribution. The Chi-square tests are conducted on all tasks of the survey dataset and the results are shown in Table 1. As we can see, as different significant level $\alpha$ is used for the test, the non-rejection rate of the normality hypothesis (i.e., the percentage of tasks for which the null hypothesis cannot be rejected) is consistently around 90%. The results further provide evidence that the random observations for the majority of tasks follow normal distribution.

![Diagram of expertise model](image)

**Fig. 1: An overview of the ETA$^2$ approach**

![Graph of non-rejection rate](image)

**TABLE 1: Non-Rejection Rate of the Chi-Square Normality Test**

<table>
<thead>
<tr>
<th>Significance Level</th>
<th>$\alpha = 0.5$</th>
<th>$\alpha = 0.25$</th>
<th>$\alpha = 0.1$</th>
<th>$\alpha = 0.05$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass Rate</td>
<td>87.18%</td>
<td>88.46%</td>
<td>89.74%</td>
<td>89.74%</td>
</tr>
</tbody>
</table>

Since the result shown in Figure 2 is accumulated from the observations for all tasks. We further verify if users’
existing techniques [10][11] to identify the expertise domains or topics of documents by statistically analyzing the appearance of words in the documents. However, these techniques cannot be directly applied to the task descriptions, because they require the documents to be long enough for effective statistical results, but the task descriptions are usually short in crowdsourcing.

To address this problem, we design a semantic analysis method, called pair-word, to extract and quantify the semantic information of the tasks based on the task descriptions. With the extracted semantic information, we are able to measure the distance (or similarity) between tasks. Then, we can cluster these tasks based on their distance so that each cluster represents one expertise domain and the tasks inside this cluster belong to the corresponding expertise domain. Specifically, a dynamic hierarchical clustering approach is proposed to cluster tasks and identify expertise. As new tasks arrive, the dynamic hierarchical clustering approach can dynamically identify their expertise domains by creating new clusters or merging to existing clusters. In the rest of the section, we first discuss how to extract the semantic information with our pair-word method, and then present the dynamic hierarchical clustering approach.

3.2 Semantic Information Extraction

To effectively extract the semantic information from the task description, we design a pair-word based method to identify two types of important terms within each description sentence: Query term, which refers to the words or phrases that describe the requirement of a specific task, and Target term, which contains the desired information. For example, the following shows two tasks and their identified Query and Target terms (Query and Target terms are manually identified).

- Task 1: What is the noise level around the municipal building?
  Query: noise level; Target: municipal building
- Task 2: How many students have attended the seminar today?
  Query: students; Target: seminar

We utilize distributed semantics of $<\text{Query}, \text{Target}>$ to capture the meaning of each task description. Word embedding is an efficient technique to map each word or phrase to a low-dimensional vector based on their global contexts. We use the Continuous Skip-gram model [12] to learn lexical representation for each single word from the entire Wikipedia dump (August 11, 2014). For multi-word terms, a simple element-wise additive model ($V = x_1 + x_2 + \ldots + x_i$) [12] is exploited, where $V$ represents a phrase embedding and $x_1, x_2, \ldots, x_i$ represent the individual embeddings of the words in $V$. We concatenate the vector representation of Query term $V_Q$ and Target term $V_T$ for each task description and use Euclidean distance metric to measure the distance between two tasks $i$ and $j$ based on their semantic vectors:

$$E(i, j) = \frac{1}{2} \left[ ||V^i_Q - V^j_Q||^2 + ||V^j_T - V^i_T||^2 \right]$$

where $[V^i_Q, V^j_T]$ denotes the concatenation of two vectors $V^i_Q$ and $V^j_T$ for each task. With this pair-word extraction method, we can efficiently capture the semantic information that two tasks shared.

3.3 Dynamic Hierarchical Clustering

3.3.1 Hierarchical Clustering

Based on the distance metric between tasks, tasks are clustered together. Although there are many clustering techniques in the literature [13], we select hierarchical clustering based on the following two reasons. First, with hierarchical clustering, the number of clusters is not fixed. As a result, clusters can be updated and new clusters can be added when new tasks are added. Second, hierarchical clustering is effective and simple with only one parameter $\gamma$, which is used to quantify the minimum allowed distance between clusters. Here, the cluster distance is calculated as the average distance between tasks in the two clusters. After the hierarchical clustering process, the distance between any two clusters should be equal or larger than the minimum allowed distance. Assume the longest distance between all existing tasks is $d^*$. The minimum allowed distance can be represented as $\gamma \cdot d^*$, where $d^*$ is a fixed value and $\gamma \in [0, 1]$ is the parameter, which can be flexibly set according to specific requirements.

With a set of $m$ tasks from the warm-up period, the basic hierarchical clustering works as follows:

1. Initialization: Each of the $m$ tasks starts its own cluster.
2. Merging clusters: Pick two clusters that are closest and merge them. This step repeats until the termination criterion is satisfied, as defined in the next step.
3. Termination: The algorithm terminates if the distance between the closest clusters in one round is equal to or larger than the minimum allowed distance between clusters, $\gamma \cdot d^*$.

3.3.2 Dynamic Hierarchical Clustering

The above algorithm can be directly applied to identify the expertise domains of the tasks in the warm-up period. As new tasks are created, they should be classified to some existing clusters, and the dynamic hierarchical clustering method is proposed to achieve this goal.

Dynamic hierarchical clustering only differs from hierarchical clustering in the initialization step. Assume $m'$ new tasks are created. For each new task, a new cluster is created, and then $m'$ new clusters are created. If there are $M$ existing clusters before new tasks are created, there will be $M + m'$ clusters in the initialization step. Then, the $M + m'$ clusters are merged following the same “Merging clusters” process, and it terminates when the termination criterion is satisfied.

4 Expertise-Aware Truth Analysis

In this section, we present our expertise-aware truth analysis solution. Specifically, a statistical model is built based on the expertise models, which treats the truth associated with each task and user expertise as parameters. By applying the technique of Maximum Likelihood Estimation (MLE), both the truth and user expertise in the statistical model can be estimated. We first present the statistical models and the MLE method used to infer the truth and user expertise. Then, we discuss how to dynamically update user expertise when new observations are made from tasks in subsequent time steps.
4.1 Estimation of Truth and User Expertise

Based on the collected data for the tasks in the warm-up period, we can find the user expertise and the truth associated with each task using MLE. With MLE, the unknown parameters of a statistical model can be estimated given the observed data. In our statistical model, the observed data is the data provided by the users for the sensing tasks. The set of observed data is denoted as

\[ X = \{X_1, X_2, ..., X_m\}, \]

where \( m \) is the number of tasks, and \( X_j \) is the set of data provided for task \( j \). The unknown parameters of our statistical model include the expertise of each user \( i \) in all of the \( D \) domains, i.e., \( u_{ij}^1, u_{ij}^2, ..., u_{ij}^m \), the ground truth for each task \( j \), i.e., \( \mu_j \) and \( \sigma_j \), and the base number for each task \( j \), i.e., \( \sigma_j \). Overall, the set of unknown parameters is represented as

\[ \Theta = \{ \mu_j, \sigma_j, \omega_{ij} \}. \]

We use \( \omega_{ij} \in \{0, 1\} \) to denote whether user \( i \) has provided data for task \( j \). If \( \omega_{ij} = 1 \), user \( i \) has provided data for task \( j \), and the data is denoted as \( x_{ij} \); otherwise, \( \omega_{ij} = 0 \).

If \( \omega_{ij} = 1 \), the probability density function (pdf) of \( x_{ij} \) is

\[ f(x_{ij} | \Theta) = \frac{1}{\sigma_j/u_{ij}^2 \sqrt{2\pi}} e^{-\left(\frac{x_{ij} - \mu_j}{2\sigma_j/u_{ij}}\right)^2}. \]

From the above equation, we can compute the pdf (or likelihood function) that \( X \) is observed as

\[ f(X | \Theta) = \prod_{j=1}^m f(X_j | \Theta) = \prod_{j=1}^m \prod_{i=1}^n (f(x_{ij} | \Theta))^\omega_{ij} \]

\[ = \prod_{j=1}^m \prod_{i=1}^n \left( \frac{1}{\sigma_j/u_{ij}^2 \sqrt{2\pi}} e^{-\left(\frac{(x_{ij} - \mu_j)^2}{2\sigma_j^2/u_{ij}^2}\right)} \right)^\omega_{ij}. \]

With \( \omega_{ij} \) being the exponent of \( f(x_{ij} | \Theta) \), \( f(X | \Theta) \) only multiplies \( f(x_{ij} | \Theta) \) with \( \omega_{ij} = 1 \).

Given the likelihood function, MLE estimates the parameters by computing the parameter set \( \hat{\Theta} \) that maximize the likelihood function. Maximizing the likelihood function is equal to maximizing the log-likelihood function, which is

\[ \log L(\Theta; X) = \log f(X | \Theta) \]

\[ = \sum_{i=1}^n \sum_{j=1}^m \omega_{ij} \left[ \log \left( \frac{1}{\sigma_j/u_{ij}^2 \sqrt{2\pi}} \right) - \frac{(x_{ij} - \mu_j)^2}{2\sigma_j^2/u_{ij}^2} \right] \]

Then, \( \hat{\Theta} \) is computed by setting the derivatives of the log-likelihood function \( \log L(\Theta; X) \) over each parameter to 0. With some derivation, we get

\[ \mu_j = \frac{\sum_{i=1}^n \omega_{ij} u_{ij} x_{ij}^2}{\sum_{i=1}^n \omega_{ij} u_{ij}^2}, \]

\[ \sigma_j = \frac{\left( \sum_{i=1}^n \omega_{ij} x_{ij}^2 - \mu_j^2 \sum_{i=1}^n \omega_{ij} u_{ij}^2 \right)}{\sum_{i=1}^n \omega_{ij}} \]

\[ u_{ij}^k = \left( \sum_{j=1}^m I(d_j = k) \omega_{ij} \right)^{-\frac{1}{2}}, \]

where \( i \in \{1, 2, ..., n\} \), \( j \in \{1, 2, ..., m\} \) and \( k \in \{1, 2, ..., D\} \). In Equation 6, \( I(d_j = k) = 1 \) if \( d_j = k \) is true, and \( I(d_j = k) = 0 \) otherwise. Though it is hard to get a close-form solution for each of the parameters, we can get the estimation of the parameters by iteratively computing the values based on Equations 5 and 6. To start the iterative process, we first set the initial values for the user expertise to be \( 1 (u_{ij}^k = 1, \forall i, k) \), and then use the three equations to iteratively compute the values of \( \mu_j \) and \( \sigma_j \), and \( u_{ij}^k \) until the calculated values converge. Here, we consider the process “converges” when the changes of users’ truth estimates are all less than 5% in two adjacent iterations.

4.2 Dynamic Update of User Expertise

With the aforementioned method, user expertise is estimated based on the observations of the existing tasks. After new tasks are created and clustered to an expertise domain, user expertise in that domain should be updated based on new observations.

Specifically, user expertise is updated based on Equation 6. For the part inside the parentheses, which is \( u_{ij}^k \), we maintain the numerator, denoted as \( N(u_{ij}^k) \), and the denominator, denoted as \( D(u_{ij}^k) \), respectively. Suppose the current time step is \( T \), and the \( t \) is the length of one time step. After new tasks from expertise domain \( k \) are finished by user \( i \) during the new time step, \( N(u_{ij}^k) \) and \( D(u_{ij}^k) \) are updated as follows:

\[ N(u_{ij}^k)^{T+t} = \alpha \cdot N(u_{ij}^k)^T + \sum_{j=1}^{m'} I(d_j = k) \omega_{ij}, \]

\[ D(u_{ij}^k)^{T+t} = \alpha \cdot D(u_{ij}^k)^T + \sum_{j=1}^{m'} I(d_j = k) \omega_{ij} (x_{ij} - \mu_j)^2 / \sigma_j^2, \]

where \( \alpha \in [0, 1] \) is the decaying factor placed on the original value to undermine the influence of the historical tasks, and \( m' \) is the number of tasks created in the current time step. Then the user expertise is updated based on \( N(u_{ij}^k)^{T+t} \) and \( D(u_{ij}^k)^{T+t} \):

\[ u_{ij}^k = \left( \frac{N(u_{ij}^k)^{T+t}}{D(u_{ij}^k)^{T+t}} \right)^{1/2}. \]

When calculating \( D(u_{ij}^k)^{T+t} \) with Equation 8, the true value \( \mu_j \) and base number \( \sigma_j \) for new task \( j \) are unknown a priori. To address this problem, \( \mu_j \) and \( \sigma_j \) are first estimated using Equations 5, in which the user expertise is initialized to the original values in time \( T \). Since the values of \( \mu_j \) and \( \sigma_j \) computed from Equations 5 may be changed after the user expertise is updated based on Equation 9, we apply the same iterative process to update \( \mu_j \), \( \sigma_j \), and \( (u_{ij}^k)^{T+t} \) until they converge.

As new tasks are created, in addition to those added to the existing expertise domains, there are also two other special cases. First, some tasks may form a new expertise domain. The user expertise in the new domain, the truth, and the base number of the new tasks in this domain are estimated using Equations 5–6. Second, by adding new tasks, it is also possible that two existing expertise domains \( k_1 \) and \( k_2 \) are merged if the new tasks are close to both \( k_1 \) and \( k_2 \). In this case, the user expertise in domain \( k_1 \) is updated by incorporating tasks in \( k_2 \), and \( k_2 \) is deleted. The users’ expertise in domain \( k_1 \) is then recalculated according to Equation 6 and Equation 9 by further including the new tasks.
5 Expertise-Aware Task Allocation

A task should be allocated to users with high expertise for that task. However, a user with high expertise in multiple domains may be allocated with many tasks and the user cannot finish all due to the limited processing capability, i.e., only limited time per day is available for completing tasks. Moreover, allocating tasks to users may generate cost (e.g., payment to users) to the server. Then, the cost should be minimized while allocating tasks to high-expertise users.

Considering these challenges, we formalize and solve two problems. The first is the max-quality task allocation problem, which maximizes the probability for tasks to be allocated to users with high expertise while ensuring the workload does not exceed the processing capability of each user. This problem improves data quality, but neglects the high-cost of task allocation. Therefore, it is suitable for cases where cost is not a concern. When cost is important, we consider the second problem of min-cost task allocation which minimizes the cost of task allocation while ensuring high-quality data are collected.

In the rest of the paper, the overall approach is still represented as ETA² if max-quality task allocation is used, and represented as ETA²-mc if min-cost task allocation is used.

5.1 Max-Quality Task Allocation

For max-quality task allocation, we first formalize the optimization problem. By proving the optimization problem to be NP-hard, we further propose a heuristic based algorithm as a solution.

5.1.1 The Max-Quality Optimization Problem

**a) Objective function:** We first compute the probability that at least one user can provide accurate data for task $j$:

$$p_j = 1 - \prod_{i=1}^{n} (1 - p_{ij})$$

(10)

where $p_{ij}$ is the probability that user $i$ can provide accurate data for task $j$. We consider the observed data to be accurate if its normalized error is smaller than $\epsilon$, where the normalized error is computed as the error to the ground truth divided by the base number. $\epsilon$ is a small constant and set to 0.1 in the paper. Then, the probability that user $i$ can provide accurate data for task $j$ can be computed as follows:

$$p_{ij} = P\left(\frac{|x_{ij} - \mu_{ij}|}{\sigma_j} < \epsilon\right) = \Phi(\epsilon u_{ij}) - \Phi(-\epsilon u_{ij})$$

(11)

The objective function is computed as the sum of the probability that at least one user can provide accurate data for each task, which is as follows:

$$\sum_{j=1}^{m} p_j = \sum_{j=1}^{m} \left[1 - \prod_{i=1}^{n} (1 - p_{ij})\right]$$

$$= \sum_{j=1}^{m} \left[1 - \prod_{i=1}^{n} (1 - \Phi(\epsilon u_{ij}) + \Phi(-\epsilon u_{ij}))\right]$$

(12)

By maximizing the objective function, the users with high expertise are selected with high priority, since users with high expertise are more likely to provide accurate information.

**b) Constraints:** The processing capability of each user $i$ is denoted as $T_i$, which is the available time for user $i$ to spend on processing tasks. The processing time for each task $j$ is denoted as $t_j$, $s_{ij} \in \{0, 1\}$ is used to denote whether task $j$ will be allocated to user $i$. Then, the constraints on the processing capability at each user can be represented as:

$$\sum_{j=1}^{m} t_j \cdot s_{ij} < T_i, \ \forall i \in \{1, 2, ..., n\}$$

(13)

Thus, the optimization problem can be formalized as follows:

$$\max \sum_{j=1}^{m} \left[1 - \prod_{i=1}^{n} (1 - \Phi(\epsilon u_{ij}) + \Phi(-\epsilon u_{ij}))^{s_{ij}}\right]$$

s.t. $\sum_{j=1}^{m} t_j \cdot s_{ij} < T_i, \ \forall i \in \{1, ..., n\}$

$$s_{ij} \in \{0, 1\}, \ \forall i \in \{1, ..., n\}, j \in \{1, ..., m\}$$

(14)

The optimization problem is proved to be NP-hard. The proof is given as follows.

**Proof.** First consider the case where there is only one user $i$. The objective function of the optimization problem becomes:

$$\sum_{j=1}^{m} \left[1 - \Phi(\epsilon u_{ij}) + \Phi(-\epsilon u_{ij})\right]^{s_{ij}}$$

Here, $1 - \Phi(\epsilon u_{ij}) + \Phi(-\epsilon u_{ij})$ is equal to $\Phi(\epsilon u_{ij}) - \Phi(-\epsilon u_{ij})$ as $s_{ij} = 1$, and equal to 0 as $s_{ij} = 0$. Therefore, $1 - (\Phi(\epsilon u_{ij}) + \Phi(-\epsilon u_{ij}))^{s_{ij}}$ is equivalent to $\Phi(\epsilon u_{ij}) - \Phi(-\epsilon u_{ij}) + s_{ij}$, and the objective function can also be written as $\sum_{j=1}^{m} (\Phi(\epsilon u_{ij}) - \Phi(-\epsilon u_{ij})) + s_{ij}$.

The optimization problem for the case where there is only one user $i$ becomes

$$\max \sum_{j=1}^{m} (\Phi(\epsilon u_{ij}) - \Phi(-\epsilon u_{ij})) + s_{ij}$$

s.t. $\sum_{j=1}^{m} t_j \cdot s_{ij} < T_i$

$$s_{ij} \in \{0, 1\}, \ \forall j \in \{1, ..., m\}$$

(15)

The above formulation is exactly the formulation of the knapsack problem, where the user $i$ with limited processing capability $T_i$ is equivalent to the knapsack with weight capacity $T_i$, and the tasks to be assigned are equivalent to the items to be put to the knapsack, each with weight $t_j$. Each item $j$ generates a value $(\Phi(\epsilon u_{ij}) - \Phi(-\epsilon u_{ij}))$ if it is placed to the knapsack. And the objective is to maximize the generated value with the limitation that the placed items do not exceed the weight capacity of the knapsack.

Since the knapsack problem is well-known to be an NP-hard problem [14], the optimization problem with single user is NP-hard. The optimization problem with single user is a reduction from the optimization problem with $n$ users, $n \geq 1$. Therefore, the optimization problem with $n$ users is also NP-hard.
5.1.2 Heuristic Based Algorithm

Since the proposed optimization problem is NP-hard, we propose a heuristic based algorithm. The basic idea is to greedily select user-task pairs which can add more value to the objective function while ensuring the tasks consume less processing time. For each selected user-task pair, the task will be assigned to that user.

In the heuristic based algorithm, we first define a concept of efficiency for each user-task pair (denoted as efficiency(i, j) for user i and task j) to quantify its importance in increasing the value of the objective function. Let \( T'_i \) denote the remaining processing capability of user i. The efficiency concept is formally defined as follows:

**Definition 1.** The efficiency of user-task pair \((i, j)\) is calculated as the value increase of the objective function divided by the processing time \( t_j \) of task j if task j is assigned to user i. However, if the remaining processing capability \( T'_i \) of user i is not enough to finish task j, i.e., \( t_j > T'_i \), the efficiency is set to zero.

Specifically, the value increase of the objective function is computed as

\[
p^{\text{new}}_j - p_j = \left[ 1 - (1 - p_j)(1 - p_{ij}) \right] - p_j = p_{ij}(1 - p_j), \tag{16}
\]

where \( p^{\text{new}}_j \) is the updated \( p_j \) after user i is added. Note that adding a user for task j does not change other tasks, so we do not need to consider other tasks in the objective function. As a result, the efficiency of \((i, j)\) is computed as:

\[
\text{efficiency}(i, j) = \begin{cases} 
\frac{p^{\text{new}}_j - p_j}{t_j}, & \text{if } T'_i \geq t_j \\
0, & \text{if } T'_i < t_j 
\end{cases} \tag{17}
\]

The heuristic based algorithm works by greedily adding the user-task pair \((i, j)\) which achieves the maximum efficiency. The greedy process terminates when the maximum efficiency that can be achieved becomes zero, which means the users have used up all their processing capabilities. The general flow of the heuristic algorithm is outlined in Algorithm 1. First, the algorithm computes efficiency\((i, j)\) for all user-task pair (Line 2). At the same time, the algorithm maintains the max efficiency \( \text{maxeff} \), that can be achieved for each task \( j \), and the user that achieves \( \text{maxeff} \), denoted as \( \text{user}_j \) (Lines 3 ~ 6), so that when the greedy algorithm selects the maximum efficiency in each round, it can simply select from the maintained \( \text{maxeff} \) for the \( m \) tasks (Line 8). Then, the algorithm greedily selects the user-task pair that achieves the maximum efficiency and updates the efficiency for other unselected pairs (Lines 8 ~ 16). The algorithm terminates when the max efficiency is equal to zero (Lines 10 ~ 12).

To select each user-task pair, \( O(m) \) time is used to find the pair with the highest efficiency. After selecting the user-task pair, \( O(m + n) \) time is used to update the efficiency of the remaining unselected user-task pairs, because only the pairs associated with the selected user or the selected task need to be updated. As a result, the total time used for selecting each user-task pair is \( O(m) + O(m + n) = O(m + n) \). Assuming \( K \) user-task pairs are selected, the time complexity of the whole process is \( O(K(m + n)) \).

**Analysis on Approximation Ratio:** The objective function of the proposed optimization problem can be proven to be submodular and monotone (the proof can be found in Appendix). Also, the constraint of the optimization problem is considered to be the type of knapsack constraint, with overall processing time (i.e., sizes) of assigned tasks not exceeding the processing capability (i.e., knapsack capacity) of users. To maximize such monotone submodular function with knapsack constraint, the proposed greedy algorithm (Algorithm 1) can usually be used as an approximate solution, especially if there is no huge difference on the processing time of tasks. However, as shown in the literature [15], if there is huge difference on the processing time of tasks, the greedy algorithm can perform arbitrarily poorly and there is no guaranteed approximation ratio for this algorithm.

To address this problem, [15] also mentions that if an extra step is added to the greedy algorithm, a guaranteed approximation ratio can still be achieved. In specific, in the end of Algorithm 1, we do another similar greedy algorithm that ignores the processing time of tasks, i.e., treating processing time of tasks as equal when calculating efficiency, and greedily selecting the user-task pair that gives the max value increase of the objective function, until exceeding users’ processing capabilities. Then, the solutions of the two greedy algorithms are compared, and the solution that achieves higher value in objective function is selected as the final task allocation solution. By doing so, the algorithm can achieve a guaranteed \( \frac{1}{2} \) approximation ratio.

Hereafter, to ensure good performance, especially in case that there exists huge difference on the processing time of tasks, the extra step as we depicted here is always added to the end of Algorithm 1.

5.2 Min-Cost Task Allocation

Task allocation involves user recruiting, which usually incurs non-negligible cost on the server side. For example, each user may be paid a specific amount \( c_j \) if the user is recruited for task \( j \) and can finish the task on time. Then, the solution of max-quality task allocation may generate a significant cost on task allocation, because it greedily allocates tasks to users within users’ processing capability, and tasks may be allocated to more users than needed while generating much higher cost. Therefore, we further formalize the min-cost task allocation problem which aims...
to minimize the cost of recruiting users while ensuring the collected data still have high quality.

5.2.1 The Min-Cost Optimization Problem

**Definition 2. Min-Cost Task Allocation:** using minimum cost for task allocation (i.e. user recruiting) so that data collected from users satisfy specific quality requirement.

The min-cost task allocation problem can be formalized as the following optimization problem:

\[
\text{min} \sum_{i=1}^{n} \sum_{j=1}^{m} s_{ij} \cdot c_j \tag{18}
\]

\[
s.t. \sum_{j=1}^{m} t_j \cdot s_{ij} < T_i, \quad \forall i \in \{1, 2, ..., n\}
\]

\[
|\hat{\mu}_j - \mu_j| < \epsilon, \quad \forall j \in \{1, 2, ..., m\} \tag{19}
\]

The objective of this optimization problem is to minimize the cost of task allocation (Formula (18)), where \(s_{ij} \in \{0, 1\}\) represents whether user \(i\) is selected for task \(j\). The first constraint in (19) ensures that the tasks assigned to user \(i\) do not exceed the processing capability of user \(i\). The second constraint in (19) ensures that, for each task \(j\), the data collected from selected users satisfy the specified quality requirement, i.e., limiting the estimation error \(|\hat{\mu}_j - \mu_j| / \sigma_j\) of each task \(j\) to be smaller than a maximum error limit \(\epsilon\). Here, \(\hat{\mu}_j\) is the estimated truth based on the data collected from selected users (all of user \(i\) with \(s_{ij} = 1\)). In this constraint, the decision variables \(s_{ij}\) are not involved explicitly, but used to decide \(\hat{\mu}_j\).

5.2.2 An Iterative Task Allocation Approach

It is a challenge to solve the min-cost optimization problem. This is because at the time of task allocation, no data has been collected yet. Therefore, it is infeasible to evaluate the estimation error and determine whether the quality requirement is satisfied.

To address this problem, we apply an iterative task allocation approach instead of allocating each task to a group of users at once. Specifically, in each iteration, each task is only allocated to a limited group of users, and the data quality is probabilistically evaluated based on the collected data. The iterative process continues allocating tasks to more users until the specified quality requirement is reached.

Figure 3 describes the iterative approach. In each iteration, the server first allocates tasks to a group of users, and collects data from these users. Then, the truth is estimated based on the expertise-aware truth analysis approach presented in Section 4. Afterwards, the server evaluates the data quality in a probabilistic manner and checks if the quality requirement can be satisfied to some confidence level. If so, the iterative process ends. Otherwise, the server starts a new iteration and allocates tasks to another group of users.

In each iteration, there are three questions to be addressed: 1) how to allocate tasks to users, 2) how to estimate truth based on the collected data, and 3) how to evaluate the data quality probabilistically. Below, we propose solutions for these three questions. The process of min-cost task allocation is outlined in Algorithm 2.

**a) Task Allocation and Truth Analysis in one Iteration:**
To minimize the cost for task allocation, we limit the cost to use in each iteration. Specifically, the cost in each iteration is limited by \(c^o\), which is a pre-defined parameter and can be set flexibly. The problem becomes allocating tasks to users that are more likely to provide high-quality data, while limiting the cost for task allocation. To address this problem, we cannot directly apply the expertise-ware task allocation method proposed in Algorithm 1, since this algorithm does not constraint the cost spent on task allocation. Here we add the cost constraint to Algorithm 1, i.e., the total cost for task allocation cannot exceed the cost limit \(c^o\). Specifically, in each iteration, the server exploits Algorithm 1 to greedily allocate tasks to users until exceeding users’ processing capabilities, or reaching the limit on task allocation cost \(c^o\) (Lines 4 ~ 7 in Algorithm 2). Note that, to ensure good performance of Algorithm 1, the extra step as we discussed in the end of Section 5.1.2 can also be added after line 7. But to be concise, it is not shown here.

After tasks are allocated to users, users collect data and send data back to server. To estimate the truth from the collected data, the expertise-aware truth analysis approach presented in Section 4 can be applied here (Lines 8 ~ 9 in Algorithm 2). But it should be noted that, when estimating truth for each task in one iteration, all the data collected for the task, including the data in the current iteration and all previous iterations, should be used.

**b) Data Quality Evaluation in one Iteration:** After the truth is estimated based on the collected data, it is still impossible to directly know whether the data quality requirement is satisfied or not, because the ground truth is

---

**Algorithm 2 Min-Cost Task Allocation**

**Input:** \(u_k, k; d_j, t_j, c_j, \forall j; c_0, \epsilon\)

**Output:** \(s'_{ij}, \forall i, j\)

1. **Initialize:** \(s_{ij} \leftarrow 0, \forall i, j\)
2. **while** true **do**
3. \(s'_{ij} \leftarrow 0, \forall i, j / *s'_{ij}\) is the new allocation in current iteration*/
4. **while** \(\sum_{i=1}^{n} \sum_{j=1}^{m} s'_{ij} \cdot c_j < c_0\) **do**
5. **Allocate a task** to a user \(i\) according to Algorithm 1
6. \(s_{ij} \leftarrow 1 \text{ and } s'_{ij} \leftarrow 1\)
7. **end while**
8. **Allocate tasks to users** according to \(s'_{ij}\) and collect data
9. **Estimate** truth of tasks \(\mu_j, \forall j\) **according to Equations** (5)–(6)
10. **pass** \(\leftarrow true\)
11. **for** Each task \(j \in \{1, 2, ..., m\}\) **do**
12. **Evaluate** confidence interval for \(\mu_j\) according to (24)
13. **if** length of confidence interval > \(2\epsilon\) **then**
14. **pass** \(\leftarrow false,\) **break**
15. **end if**
16. **end for**
17. **if** pass = true **then**
18. **break**
19. **end if**
20. **end while**
unknown. To address this problem, we evaluate the data quality in a probabilistic manner, and ensure that the specified quality requirement can be satisfied to some confidence level.

For each task \( j \), instead of directly constraining the estimation error by \( \bar{\varepsilon} \), i.e., \( \frac{|\mu_j - \hat{\mu}_j|}{\sigma_j} \leq \bar{\varepsilon} \), we ensure the normalized error to be smaller than \( \bar{\varepsilon} \) with confidence \( 1 - \alpha \), i.e.,

\[
P\left( \frac{|\mu_j - \hat{\mu}_j|}{\sigma_j} \leq \bar{\varepsilon} \right) > 1 - \alpha.
\]

Here, \( \alpha \) is a small probability, and is usually set to 0.05 if the 95\% confidence is required, or set to 0.1 if the 90\% confidence is required. The inequality in (20) is equivalent to:

\[
P(\mu_j \in [\hat{\mu}_j - \bar{\varepsilon}\sigma_j, \hat{\mu}_j + \bar{\varepsilon}\sigma_j]) > 1 - \alpha
\]

In this formula, \( [\hat{\mu}_j - \bar{\varepsilon}\sigma_j, \hat{\mu}_j + \bar{\varepsilon}\sigma_j] \) is in fact the \( 1 - \alpha \) confidence interval for the ground truth \( \mu_j \). As long as we find the \( 1 - \alpha \) confidence interval for \( \mu_j \) is smaller than \( [\hat{\mu}_j - \bar{\varepsilon}\sigma_j, \hat{\mu}_j + \bar{\varepsilon}\sigma_j] \), the inequality in formula (21) can be satisfied, so that the requirement on data quality can be satisfied with confidence \( 1 - \alpha \).

The confidence interval for \( \mu_j \) can be approximately calculated based on one of the asymptotic properties of MLE, i.e., asymptotic normality [16]:

**Theorem 1. Asymptotic Normality:** Distribution of MLE estimators for a parameter \( \theta \) is asymptotically normal with mean \( \theta \) and variance \( \text{var}(\hat{\theta}) \), which can be approximated by the inverse of the Fisher information \( I(\theta) \):

\[
I(\theta) = E_\theta(\frac{\partial}{\partial \theta} \log f(X|\theta))^2 = -E_\theta(\frac{\partial^2}{\partial \theta^2} \log f(X|\theta))
\]

(22)

According to asymptotic normality, the MLE estimator \( \hat{\mu}_j \) is asymptotically normal with mean \( \mu_j \), and its variance \( \text{var}(\hat{\mu}_j) \) is approximated by:

\[
\frac{1}{I(\mu_j)} = \frac{1}{-E_{\mu_j}(\frac{\partial}{\partial \mu_j} \log f(X|\mu_j))^2} = \frac{\sigma_j^2}{\sum_{i=1}^n s_{ij}(u_{ij}^j)^2}
\]

(23)

Based on the property of normal distribution, the \( 1 - \alpha \) confidence interval for \( \mu_j \) is,

\[
[\hat{\mu}_j - Z_{\alpha/2} \frac{1}{\sqrt{I(\mu_j)}} , \hat{\mu}_j + Z_{\alpha/2} \frac{1}{\sqrt{I(\mu_j)}}] = [\hat{\mu}_j - Z_{\alpha/2} \frac{\sigma_j}{\sqrt{\sum_{i=1}^n s_{ij}(u_{ij}^j)^2}} , \hat{\mu}_j + Z_{\alpha/2} \frac{\sigma_j}{\sqrt{\sum_{i=1}^n s_{ij}(u_{ij}^j)^2}}]
\]

(24)

where \( Z_{\alpha/2} \) is the \( \alpha/2 \) quantile of the standard normal distribution. As long as the \( 1 - \alpha \) confidence interval for \( \mu_j \) calculated in (24) is smaller than \( [\hat{\mu}_j - \bar{\varepsilon}\sigma_j, \hat{\mu}_j + \bar{\varepsilon}\sigma_j] \), i.e., the length of confidence interval is shorter than \( 2\bar{\varepsilon}\sigma_j \), the quality requirement can be satisfied. Otherwise, the server will start another iteration. The process of evaluating data quality for each task corresponds to Lines 10 ~ 19 in Algorithm 2.

### 6 Performance Evaluations

In this section, we evaluate the performance of ETA\(^2\) by conducting experiments based on two real-world datasets and a synthetic dataset.

#### 6.1 Datasets

##### 6.1.1 Survey-based Dataset

The first real-world dataset is collected through a survey including 60 participants on our campus, after IRB approval. In the survey, each of the participants is required to answer 89 questions about their daily life and basic knowledge on various topics. Some sample questions are listed as follows:

- How many parking lots on campus are open to students in this semester?
- What is the estimated driving hours to another city in the local state?
- What is the average salary for an entry-level software engineers in United States?

The answers to some questions may depend on specific time and location. For example, the available parking lots to students may be different during weekdays and weekends, and the driving hours may be different during early morning and late afternoon in each day. Therefore, some questions are replicated to consider the conditions at different time and locations. As a result, there are finally 150 questions in the dataset.

The provided answers are noisy, and then can be used for evaluating our approaches. Assume these questions are the sensing tasks provided by the server, and the content of the question is the task description. If a user is queried with a sensing task, the collected data value is the answer to the corresponding question. For the purpose of performance evaluation, the ground truth of every question is carefully inspected and added by the researchers.

##### 6.1.2 SFV Dataset

The second real-world dataset [17] is extracted from the Slot-Filling Validation (SFV) task of the 2013 Text Analysis Conference (TAC) Knowledge Base Population (KBP) track. In the task, 18 slot-filling systems are required to answer a set of questions about 100 entities, including famous persons or organizations. The questions are about various properties of the entities, like the age, birthday and name. There are about 2,000 questions for these 100 entities in the original dataset. Similar to the survey-based dataset, each question is treated as a sensing task. The 18 slot-filling systems are treated as users, and their answers to the questions are treated as the collected sensing data. In the dataset, documents that describe the entities and their properties are also given, from which we can easily compose the descriptions of tasks. The ground truth of the tasks is also included in the original dataset, which can be used for performance evaluations.

##### 6.1.3 Synthetic Dataset

The above two real-world datasets enable us to test our approaches in real-world environments. For example, with the ground truth given in both two datasets, we can evaluate the estimation errors of ETA\(^2\). However, some other critical values, e.g., the user expertise, are not given in these datasets, so that we cannot evaluate whether our approach can estimate these values correctly. Given this, in addition to the two real-world datasets, we also create a synthetic dataset, so that we can thoroughly evaluate the
estimation error with different parameter settings for the three dataset.

Fig. 4: Estimation error with different parameter settings for the three dataset.

effectiveness of ETA$^2$, including its capability in estimating key parameters, like user expertise.

Without loss of generality, in the synthetic dataset we generate 100 users and 8 expertise domains. The expertise $u_{ij}$ of user $i$ in each domain $d$ is randomly generated within [0, 3]. In addition, we generate 1000 sensing tasks. The ground truth $\mu_j$ for task $j$ is randomly generated within [0, 20], and the base number $\sigma^d$ is randomly generated within [0.5, 5]. Different to the real-world tasks that usually have textual task descriptions and require the “pair-word” and hierarchical clustering approaches to identify the expertise domains, in the synthetic dataset, each task is explicitly assigned to an expertise domain $d_j$ that is pre-known to the server. For each sensing task $j$, the data value observed by user $i$, if task $j$ is allocated to user $i$, follows normal distribution $N(\mu_j, (\sigma_j/u_{ij})^2)$, where $u_{ij}$ is the expertise of user $i$ in the expertise domain of task $j$.

6.2 Experimental Setting

In these datasets, some important information like the processing time required for each task and the processing capability of each user is not yet provided. Without loss of generality, these quantities are generated using uniform distribution. For the survey-based dataset, the processing time required for each task is randomly generated between [2, 4] hours. For the SFV dataset and the synthetic dataset, it is randomly generated in [1, 2] hours and [0.5, 1.5] hours respectively. The processing capability of each user is randomly generated between [$\tau – 4$, $\tau + 4$] hours, where $\tau$ is a variable representing the average processing capability. $\tau$ is set to 12, but can be changed to evaluate how the processing capability affects the performance. The length of the time step is a day in the experiment. The sensing tasks are assumed to be generated and evenly distributed during five days.

In order to have statistically converging results, we set different seeds to randomly select tasks in each day and take the average as the results. Specifically, every experiment is run 100 times by setting different seeds to achieve statistical convergence.

6.3 Approaches in Comparison

We compare our proposed expertise-aware solution with three most well-known approaches that estimate truth by calculating source reliability, and a baseline approach serving as the lower bound. These approaches in comparison are as follows:

- **Hubs and Authorities** [18]: The reliability of a source is the sum of the credibility (i.e., quality) of the data items it provides, and the credibility of a data item is the sum of the reliability of sources that provide the data.
- **Average-Log** [5]: The reliability of each source is calculated by multiplying the average credibility of its provided data item and the logarithm of the number of its provided data item.
- **TruthFinder** [4]: The credibility of an observed data item is the probability that it is accurate and the reliability of the source is the probability that it provides accurate data. Specifically, the credibility of a data item is computed as the probability that at least one source can provide accurate data. A source’s reliability is calculated by averaging the credibility of its provided data.
- **Baseline**: The truth is estimated as the mean value of the observed data.

These approaches in comparison only specify the method for truth analysis. To allocate new tasks to users, the first three reliability-based approaches greedily allocate tasks to users with high reliability. Considering users only have limited processing capability, we prioritize the tasks with lower sensing time to be allocated to users with high-reliability, so that these high-reliability users can finish as many tasks as possible. For the baseline approach, the tasks are allocated to users randomly.

6.4 Evaluation Results

In the evaluations, we first show how the parameters $\gamma$ and $\alpha$ affect the performance of ETA$^2$ approach. Then, we evaluate the overall performance of ETA$^2$ by comparing it with existing approaches. Third, we evaluate the effectiveness of the three modules of ETA$^2$. In these evaluations, the performance metric used is the estimation error, which is computed as the average of the normalized estimation error for all sensing tasks. In the first three evaluations, we do not consider the task allocation cost, and the experiments are based on ETA$^2$, in which the task allocation is based on max-quality task allocation. In the last evaluation (Section 6.4.3), we compare the performance of ETA$^2$ and ETA$^2$-mc in terms of both estimation error and task allocation cost.

6.4.1 The Effects of Parameters

ETA$^2$ has two parameters $\alpha$ and $\gamma$. When updating user expertise to include the new tasks, a decaying factor
α ∈ [0, 1] is placed on the historical tasks to reduce the influence of the historical tasks. Parameter γ ∈ [0, 1] is used to determine the minimum allowed distance between clusters, which determines when the merging process terminates. It is crucial to set appropriate α and γ so that the performance of our approach can be optimized. Note that the parameter γ is not used in the synthetic dataset which only uses parameter α. This is because the expertise domains of the tasks are pre-known. Then, there is no need to use clustering to identify the expertise domains, and there is no need to use parameter γ which is for clustering.

We conduct experiments to evaluate how the parameters affect the performance of ETA². The objective is to evaluate the estimation error under different parameter settings and find the set of parameters which result in best performance. The results for the three datasets are shown in Figure 4 (a)-(c), respectively. In Figure 4 (a) and (b), the z axis is upside down for better visualization, so that the point with the smallest estimation error is shown in the upmost place. As we can see, the best performance can be achieved when α = 0.5 and γ = 0.6 for the survey-based dataset, and α = 0.1 and γ = 0.5 for the SFV dataset. From Figure 4 (c) we can see, the estimation error is the lowest when α = 0.5 for the synthetic dataset. Since the parameters for different datasets may be different, when ETA² is implemented, the parameters are first evaluated based on a pre-set warm-up period and then the chosen values are applied to the experiments. In the rest of the evaluation, α and γ will be set to these values that achieve the best performance according our evaluations.

6.4.2 Overall Performance of ETA²

We further evaluate the performance of ETA² by comparing it with existing approaches. First, we keep track of the estimation error in different days. The results for the three datasets are shown in Figure 5. As shown in the figures, the estimation error of ETA² drops overtime in all datasets. This is because in the beginning, there is no initial knowledge on user expertise, and the tasks are randomly allocated to users. After more tasks have been finished, user expertise is better estimated, which can be used in the following days to make better decisions on task allocation. Also, ETA² outperforms other approaches. Specifically, for the survey-based dataset, its estimation error is 15% to 20% lower compared with the existing approaches. For the SFV-based dataset, the estimation error is at least 5% to 15% lower compared with other approaches. And for the synthetic dataset, the estimation error is about 20% lower compared with other approaches.

Second, we change the average processing capability τ. A large τ means that users have more time to complete the tasks and therefore more users can be selected for each task. As shown in Figure 6, the estimation error decreases as the average processing capability increases. When the processing capability is small, ETA² underperforms TruthFinder in the survey dataset and Hubs and Authority in the SFV dataset. This is because there are very limited users assigned for each task, and then the user expertise cannot be accurately estimated. As the processing capability increases, ETA² significantly outperforms other approaches.

Verifying the importance of expertise: We further verify the importance of the expertise information, by conducting a small experiment to figure out how the expertise of users affects the data they observe, especially in the two real-world datasets. Specifically, the experiments measure the observation error, i.e., the error of the observed data, under different user expertise. The results are shown in Figure 7. As we can see, with the increase of user expertise, there is a clear decrease of the observation error. When the user expertise is larger than 2, most observation errors are close to zero (the red line inside the box indicates the median of observation errors). This result demonstrates that user...
expertise can be exploited to improve data quality in ETA$^2$.

**Assessment of the normality assumption for random observation:** In Section 2.3, the result of the Chi-square test implies that the random observations of users can be approximated by normal distribution. With this implication, we later formalize our problem and propose the solution of expertise-aware truth analysis based on the normality assumption. Since normality assumption for random observations may not apply in some datasets, we further conduct experiments with the synthetic dataset to verify if the framework still works when there is some bias within the distribution. In the synthetic dataset, different to the original setting, users’ observations for tasks are only partially generated from normal distribution, and the rest of observations are randomly generated from uniform distribution with the same mean value and standard deviation. In the experiment, we vary the proportion of observations not generated from normal distribution to test how our framework is sensitive to the bias. Figure 8 shows the result of the experiment. As can be seen, as there is more bias, i.e., more observations not generated from normal distribution, the estimation error is still consistently low, with only a slight increase. This result demonstrates that our framework is not very sensitive to the bias in the normal distribution.

### 6.4.3 Comparison Between ETA$^2$ and ETA$^2$-mc

The previous experiments have demonstrated the superiority of ETA$^2$ in terms of estimation error. In the following experiment, we further consider the task allocation cost, i.e., the server gives users some payment when allocating tasks. We evaluate if the ETA$^2$-mc approach can reduce the task allocation cost with comparable estimation error compared to ETA$^2$.

**Experimental Setting:** We define the task allocation cost to be one unit when one task is allocated to one user. For example, a user is paid $1 for each task he or she finishes. Therefore, $c_j$ is equal to 1 in (18), and the total cost for task allocation is equal to $\sum_{i=1}^{m} \sum_{j=1}^{s_{ij}}$. The quality requirement in ETA$^2$-mc is that the estimation error is smaller than the maximum error limit $\bar{\epsilon}$ with confidence $1 - \alpha$. Here, we set $\bar{\epsilon}$ to be 0.5, and $\alpha$ to be 0.05 to achieve the 95% confidence level. In the iterative task allocation approach, $c^o$ is the cost used for task allocation in each iteration. Here, $c^o$ is set to different values to test how it affects the performance of ETA$^2$-mc.

We compare ETA$^2$ and ETA$^2$-mc in terms of estimation error and task allocation cost. Different values of $c^o$ are tested in each dataset. In the experiment, we change the average processing capability of users and check how it affects the performance. The results are shown in Figure 9 and Figure 10. In Figure 9, we also show the quality requirement (error $< \bar{\epsilon}$, $\bar{\epsilon} = 0.5$) for reference purpose.

**Comparison between ETA$^2$ and ETA$^2$-mc:** As can be seen from these two figures, ETA$^2$-mc has similar estimation error to ETA$^2$, but with much less cost. This is especially true for the SFV dataset and the synthetic dataset, and it demonstrates that ETA$^2$-mc can effectively reduce the cost, while keeping the estimation error low. For the survey-based dataset, we can clearly see a lower estimation error of ETA$^2$ than ETA$^2$-mc, especially when users’ processing capability is high. However, ETA$^2$ has much higher cost. In many cases, using higher cost to achieve low error is unnecessary, especially when the objective is to meet some specific quality requirement (i.e. error $< \bar{\epsilon}$) with minimum task allocation cost. With this objective, ETA$^2$-mc shows its superiority, since for all three datasets, ETA$^2$-mc can successfully satisfy the quality requirement with much lower cost.

**Influence of $c^o$:** From Figure 9 and Figure 10, we also see that the change of $c^o$ does not have a major impact on the estimation error and task allocation cost. If $c^o$ is set to a proper range, it does not have significant impact on the results. It is worth to note that, setting $c^o$ to very low or very high values may reduce the performance. When $c^o$ is set too low, only limited tasks can be allocated and limited data are provided in each iteration, which may not correctly estimate the user expertise in the first several iterations. When $c^o$ is set too high, there will be too much cost in each iteration, which makes the overall cost to be unnecessarily high.

### 6.4.4 Discussions

**The accuracy of user expertise estimation:** Expertise-aware truth analysis exploits MLE to estimate user expertise as well as the truth of tasks. The previous experiments have provided enough evidence that truth can be accurately estimated (with small estimation error). We then conduct a small experiment to check if the user expertise can also be accurately estimated. We use the synthetic dataset in this experiment, because the two real-world datasets, survey-based dataset and SFV dataset, do not provide the real values of user expertise, unfortunately. In this experiment, we calculate the estimation error on user expertise using ETA$^2$. The average processing capability of users is varied to see how it impacts the expertise estimation error. The result is shown in Figure 11, from which we can see, the expertise estimation error keeps decreasing as the users’ processing capability increases. When the average processing capability
We further evaluate in how many iterations the MLE can converge based on Equation 5 and Equation 6. In the last experiment, we identify for a task, the task needs to be allocated to more users. Approximately 40% of tasks are allocated to 6 to 10 users. The highest number of users assigned to a task is 20. Further considering the average expertise of the assigned users, we find that for tasks with less users, their average expertise is usually higher. The results concur with the intuition that if a task is allocated to high-expertise users, less users are needed to provide accurate results. However, if no high-expertise users can be identified for a task, the task needs to be allocated to more users with moderate expertise, to ensure that enough data can be collected to infer accurate results. The results verify that the heuristic-based algorithm can effectively allocate tasks to sufficient number of users with right expertise.

The convergence of expertise-aware truth analysis: In expertise-aware truth analysis, MLE is used to estimate the parameters. Specifically, an iterative process is conducted based on Equation 5 and Equation 6. In the last experiment, we further evaluate in how many iterations the MLE can converge based on the three datasets. Figure 12 shows the cumulative distribution function (CDF) of the iterations needed before convergence. As we can see, the majority of the processes can converge within only 10 iterations. Almost all processes in survey and SFV datasets can converge within 20 iterations, and almost all processes in synthetic datasets can converge within 60 iterations. These results show that MLE used in expertise-aware truth analysis can converge in a fast speed in all of the three datasets.

The performance of the heuristic-based algorithm for max-quality task allocation: In expertise-aware task allocation, a heuristic-based algorithm is proposed to allocate tasks to users. To evaluate the effectiveness of task allocation, we conduct an experiment to record how many users are allocated to each task and what is the average expertise of the users in the task’s expertise domain. The results are shown in Table 2. As we can see, each task can be allocated to at least two users. Approximately 40% of tasks are allocated to 6 to 10 users. The highest number of users assigned to a task is 20. Further considering the average expertise of the assigned users, we find that for tasks with less users, their average expertise is usually higher. The results concur with the intuition that if a task is allocated to high-expertise users, less users are needed to provide accurate results. However, if no high-expertise users can be identified for a task, the task needs to be allocated to more users with moderate expertise, to ensure that enough data can be collected to infer accurate results. The results verify that the heuristic-based algorithm can effectively allocate tasks to sufficient number of users with right expertise.

<table>
<thead>
<tr>
<th>Number of users assigned</th>
<th>Tasks</th>
<th>Average expertise of users</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2, 5]</td>
<td>20.9%</td>
<td>2.57</td>
</tr>
<tr>
<td>[6, 10]</td>
<td>40.3%</td>
<td>1.85</td>
</tr>
<tr>
<td>[11, 15]</td>
<td>20.9%</td>
<td>1.37</td>
</tr>
<tr>
<td>[16, 20]</td>
<td>17.7%</td>
<td>1.27</td>
</tr>
</tbody>
</table>
7 Related Work

By taking advantage of numerous mobile users to provide valuable data about themselves and their surroundings, mobile crowdsourcing helps enable many applications such as traffic monitoring [19], urban noise mapping [20], indoor localization [21], network quality measurement [3], image sensing [2], and etc. In mobile crowdsourcing, task allocation is a very important procedure because it determines how to select appropriate users to successfully complete tasks for those applications. Researchers have designed solutions for task allocation by focusing on different perspectives. For example, the authors in [22][23] consider energy-efficient task allocation which recruits users to reduce the energy consumption while achieving the objective on spacial coverage. Authors in [24][25] focus on location-dependent mobile crowdsourcing. Wang et al. [24] propose a multi-objective optimization problem for task allocation to maximize spatial task coverage while minimizing incentive cost. Cheung et al. [25] propose a technique that enable mobile users to select tasks in a distributed manner, with the objective to collect time-sensitive and location-dependent information. Moreover, Wu et al. [2] consider image sensing with smartphones and propose user selection approach to maximize the photo utility and minimize the resource consumption. Later, Wu et al. also study the problem based on disruption-tolerant network [26] and resource-constraint environments [27]. Researchers in [8][7][9] study how the quality of data source affects the provided information and provide solutions to identify and recruit high-quality data sources.

Besides the above approaches on task allocation, some existing works [28][29] attempt to identify and recruit the high-expertise users by taking advantage of user profiles from other sources, e.g., social media sites. Those approaches are especially useful when users are directly invited from social media to perform mobile crowdsourcing tasks. By checking the users’ profiles and activities in social media, high-expertise users for tasks can be identified and invited. However, in more general scenarios where users are not recruited from social media, it is a challenge to link users to their social media accounts, unless the users agree to provide by themselves. Our work assumes the general scenarios where no information is known from other sources, and proposes techniques to self-learn user expertise from users’ previous responses.

In addition to task allocation, other research problems in mobile crowdsourcing include privacy issues and the design of incentive mechanisms. Specifically, researchers in [30][31] consider privacy issues related to the crowdsourcing participants and design privacy-preserving schemes for data collection. Researchers in [1][32][33] design incentive mechanisms and propose techniques to recruit users while minimizing cost. In addition to minimizing the cost, researchers in [34][35] propose incentive mechanisms to consider the quality of information and encourage users to provide higher-quality data by paying more to users. These mechanisms on privacy protection and incentive are orthogonal to the contributions of this paper, but can be easily built on top of our strategy.

Truth analysis in crowdsourcing has received considerable attention recently. Most truth analysis techniques estimate truth by inferring the reliability of users. One of the earliest techniques is Hubs and Authorities [18], which iteratively evaluates the correctness of information by evaluating the reliability of data sources. Pasternack et al. [5] extend these frameworks to a more general scenario by incorporating prior knowledge of the information. TruthFinder proposed in [4] is also based on an iterative method to infer the information correctness and user reliability. Researchers also propose truth analysis techniques by designing and investigating statistical models, such as bayesian inference [36] and expectation-maximization (EM) [6]. In addition to these techniques, Zhang et al. [37] propose a resource-aware truth analysis approach where data can be adaptively collected from networks to further improve data quality. Also, researchers in [38][39][40] study other issues related to truth analysis, such as truth existence, truth evolution and correlation among entities. Later, researchers further apply truth analysis to novel applications such as medical diagnosis [41] and fake news detection [42].

8 Conclusions

In this paper, we proposed ETA² approaches for expertise-aware truth analysis and task allocation in mobile crowdsourcing. Specifically, we first identified the expertise domains of the tasks by designing a novel semantic analysis method to extract the semantic information of tasks and proposing a dynamic hierarchical clustering approach to cluster tasks based on their semantic information. Then, we proposed an expertise-aware truth analysis solution, in which we built an expertise-aware statistical model and applied maximum likelihood estimation to estimate truth and learn user expertise. Finally, we designed a max-quality task allocation technique by formalizing an optimization problem to maximize the probability that tasks are allocated to users with high expertise while ensuring the work load does not exceed the processing capability at each user. Considering that task allocation has a cost, we further propose a min-cost task allocation solution (ETA²-mc) to minimize the cost of task allocation while ensuring high-quality data can be collected. Experimental results based on two real-world datasets and one synthetic dataset demonstrate that ETA² significantly outperforms existing solutions, and ETA²-mc has much lower cost than ETA².

References

Lifu Huang received a bachelor degree in 2011 from Northwestern Polytechnic University and a master degree in 2014 from Peking University. He is currently a PhD student at the School of Computer Science in Rensselaer Polytechnic Institute. His current research interests include information extraction, machine reading comprehension and machine learning.

Heng Ji is Edward P. Hamilton Chair Professor in Computer Science Department of Rensselaer Polytechnic Institute. She received her B.A. and M. A. in Computational Linguistics from Tsinghua University, and her M.S. and Ph.D. in Computer Science from New York University. Her research interests focus on Natural Language Processing and its connections with Data Mining, Social Science and Vision. She was selected as “Young Scientist” and a member of the Global Future Council on the Future of Computing by the World Economic Forum in 2016 and 2017. She received “AI's 10 to Watch” Award by IEEE Intelligent Systems in 2013, NSF CAREER award in 2009, Google Research Awards in 2009 and 2014, Sloan Junior Faculty Award in 2012, IBM Watson Faculty Award in 2012 and 2014, Bosch Research Awards in 2015, 2016 and 2017, PACLIC 2012 Best Paper Runner-up, “Best of SDM2013” paper, “Best of ICDM2013” paper.

Guohong Cao received the PhD degree in computer science from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Distinguished Professor. His research interests include wireless networks, mobile systems, wireless security and privacy, and Internet of Things. He has published more than 200 papers which have been cited over 20000 times, with an h-index of 70. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, and IEEE Transactions on Vehicular Technology, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS, MASS, and INFOCOM. He has received several best paper awards, the IEEE INFOCOM Test of Time award, and the NSF CAREER award. He is a Fellow of the IEEE.