Coverage-Oriented Task Assignment for Mobile Crowdsensing

Shiwei Song, Zhidan Liu, *Member, IEEE*, Zhenjiang Li, *Member, IEEE*, Tianzhang Xing, *Member, IEEE*, and Dingyi Fang, *Member, IEEE*

***Abstract*—Crowdsensing tasks are usually described by certain features or attributes, and the task assignment essentially performs a matching with respect to the worker or user's preference on these features. However, the existing matching strategy could lead to a misaligned task coverage problem, i.e., some popular tasks tend to enter workers' candidate task lists, while some less popular tasks could be always unsuccessfully assigned. To ensure task coverage after the assignment, the system may have to increase their biding costs to reassign such tasks, which causes a high operational cost of the crowdsensing system. To address this problem, we propose to migrate certain qualified workers to the less popular tasks for increasing the task coverage and meanwhile, optimize other performance factors. By doing this, other performance factors, such as task acceptance and quality, can be comparably achieved as recent designs, while the system cost can be largely reduced. Following this idea, this article presents *cTaskMat*, which learns and exploits workers' task preferences to achieve coverage-ensured task assignments. We implement the *cTaskMat* design and evaluate its performance using both real-world experiments and data set-driven evaluations, also with the comparison with the state-of-the-art designs.**

***Index Terms*—Mobile crowdsensing, preference, task assignment, task coverage.**

I. Introduction

Due to the increasing popularity of mobile devices, crowdsensing [7] emerges as an important technique to coordinate a group of individuals to complete certain tasks collectively. It can serve as a powerful weapon to collect a large volume of useful data to enable many useful applications. For instance, in the urban traffic engineering [18], [38], the concurrent multisource data (e.g., pedestrian mobility, bus traffic, metro statistics, etc.) need to be collected for a metropolitan-scale human mobility exploring [37]. In this case, each type of data collection can be organized as a task, and workers (i.e., users) then participate according to their choices on certain data type(s). Finally, crowdsensing helps complete the data collection from the crowd. Besides, other typical applications include environment monitoring [31], event detection [20], [21], traffic monitoring [16], social analysis [34], etc.

In general, crowdsensing assigns a set of tasks to a group of workers according to their preferences and leverages the sensing and/or computing abilities from the workers' mobiles to complete these tasks. Of course, the crowdsensing system will pay each worker as their returns [36]. During this process, one key metric to assess the performance of crowdsensing systems is the overall system cost [40]. Basically, each task can be described by certain features or attributes and each worker has her preferences to different task attributes, based on which the system can assign tasks. However, if we directly borrow the task assignment design from the recommender system domain [1], it could lead to a misaligned task coverage, i.e., some popular tasks tend to enter workers' candidate task lists (the list usually has a limited size to display the most matched tasks), while some less popular tasks could be unsuccessfully assigned. To ensure the task coverage, the system may have to increase its biding costs (cost can be one attribute) to reassign the tasks.

For those unassigned tasks, purely from the feature's matching point of view, there may already exist sufficient workers who likely accomplish them. In other words, these tasks can be potentially assigned, but the challenge is that popular tasks could lead to better matchings, e.g., higher matching scores, so as to exclude less popular tasks from the candidate lists. Thus, our key idea is to migrate certain suitable workers to those less popular tasks (still satisfying their preferences yet) for increasing the task coverage and meanwhile, also strive to optimize other performance factors. In this article, we find that with a careful coverage-oriented task assignment design, other performance factors, like the task acceptance rate and quality, can be comparably achieved as recent designs [35], while the system cost can be largely reduced, which can dramatically improve the overall benefit of the design and also make mobile crowdsensing systems much more practical.

Manuscript received December 3, 2019; revised March 8, 2020; accepted March 24, 2020. Date of publication April 1, 2020; date of current version August 12, 2020. This work was supported in part by China NSFC under Grant 61802261 and Grant 61702416, in part by the NSF Grant of Shenzhen University under Grant 2018061, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515011502, in part by the ECS Grant from Research Grants Council of Hong Kong under Grant CityU 21203516, and in part by the Shaanxi Science and Technology Innovation Team Support Project under Grant 2018TD-026. *(Corresponding author: Tianzhang Xing.)*

Shiwei Song, Tianzhang Xing, and Dingyi Fang are with the School of Information Science and Technology and Shaanxi International Joint Research Centre for the Battery-Free Internet of Things, Northwest University, Xi'an 710069, China (e-mail: shiweisong@stumail.nwu.edu.cn; xtz@nwu.edu.cn; dyf@nwu.edu.cn).

Zhidan Liu is with the Guangdong Laboratory of Artificial Intelligence and Digital Economy and College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: liuzhidan@szu.edu.cn).

Zhenjiang Li is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: zhenjiang.li@cityu.edu.hk).

Digital Object Identifier 10.1109/JIOT.2020.2984826

To this end, this article presents *cTaskMat*, with the following two main steps. To facilitate the *cTaskMat* design, in addition to worker's preferences to each potential task attributes, denoted as positive attributes, we also introduce negative attributes that indicate the worker's disliking of certain attributes.

1) We exploit and promote existing techniques to achieve an initial task assignment, which may cause the misaligned task coverage issue. Then, we leverage workers' negative attributes to exclude certain popular tasks from the candidate task lists, if possible, and supplement with less popular tasks (still satisfying their preferences). Thus, all workers' candidate lists together could achieve a good coverage for the entire task set. To this end, we propose a worker profiling and worker-attribute model-based task assignment method.
2) Rather than blindly raising task rewards to attract workers, just as the previous methods do, to assign the unsuccessfully assigned tasks, we categorize available workers according to their preferences and attitudes on rewards, and then *differentially* raise rewards of the remaining tasks according to the worker types. Such a cost-efficient remedy method not only enhances the overall task coverage but also ensures task acceptance and saves system cost.

We implement the *cTaskMat* system with the designs above and evaluate its performance extensively. We first recruit volunteers to complete a series of traffic monitoring tasks. We further leverage an existing data set [8] to understand *cTaskMat*'s performance under a larger scale setting. Compared with the state-of-the-art approaches [15], [25], [30], experimental results show that *cTaskMat* can reduce system cost by 28% at least, while sacrificing matching quality by 7% at most. In summary, the contributions of this article can be summarized as follows.

1) We identify a misaligned coverage problem in the task assignment for the mobile crowdsensing that could largely increase the system operation cost.
2) We propose a coverage-oriented solution, which formulates this problem and achieves a good tradeoff between performance-related factors and task coverage.
3) We implement the *cTaskMat* design and conduct a real-world crowdsensing experiment as well as data set-driven evaluations to examine the performance of *cTaskMat*.

The remainder of this article is organized as follows. Section II presents the problem statement and motivation. We introduce the *cTaskMat* design in Section III and evaluate its performance in Section IV. Related works are reviewed in Section V and we conclude in Section VI.

II. Problem Statement and Motivation

In this section, we introduce the task assignment problem in prior crowdsensing systems (Section II-A), analyze the misaligned task coverage problem of mobile crowdsensing through a motivation example (Section II-B), and highlight our design with *cTaskMat* (Section II-C).

TABLE I
Key Mathematical Notations Adopted in This Article

Notation	Meaning
$t_i^{c_j}$	Task i in category c_j
$a_k^{c_j}$	k-th attribute for the task category c_j
n_{max}	Maximum tasks can be undertook by a worker
n_{cand}	Number of candidate tasks for a worker to pick
$\mathcal{S}_{match}$	Matching score

A. Problem Statement

A crowdsensing system involves two major parts: 1) *tasks* and 2) *workers*. It assigns unfinished tasks to a set of registered workers by matching the attributes of tasks and the preferences or interests of workers to complete with the best matching, according to certain criteria, e.g., system cost, task acceptance, quality, latency, etc. Next, we introduce the task and worker models to facilitate our following discussions, and the key notations are summarized in Table I.

Task Model: The crowdsensing system maintains a task set $\mathcal{T}$, involving many different task category $\mathcal{C}$. Therefore, each specific task is denoted as $t_i^{c_j}$, where $i = 1, 2, \ldots, |\mathcal{T}|$ and $c_j \in \mathcal{C}$. For instance, in the multisource data collection example (stated in Section I) for the urban traffic monitoring, traffic sensing and road surface condition sensing can be two categories, where specific tasks may need to sense: bus traffic, taxi traffic, potholes, construction spots, etc.

In addition, to facilitate the task assignment, each task category has some attributes, e.g., describing its potential costs and requirements on different aspects. For each task category c_j, we denote its attributes as a vector with m items

$$\vec{\mathbf{a}}^{c_j} = <a_1^{c_j}, a_2^{c_j}, \ldots, a_m^{c_j}>$$

and each $t_i^{c_j}$ has a concrete instance of this attribute vector. In particular, the reward that pays a worker after the completion of one task is also an attribute.

Back to the urban data collection example, one task category could be "*Check the road surface condition at a specific location*," which requires the workers to travel to the specific location and then measure different types of traffics. Such tasks have potential costs on traveling distance along target roads and energy consumption of mobile devices for taking and uploading pictures. In addition, this task category is not time sensitive and could be finished within a short time with a small duration. Therefore, a possible attribute vector for this category of tasks could be <*traveling distance, urgency, duration, data type, reward*>. It is clear that these task attributes will influence the choices of workers to select tasks.

Worker Model: There are a set of human workers, denoted as $w_i \in \mathcal{W}$, who have registered to the system for accepting tasks. Each worker may prefer different categories of tasks and such preference information can be directly provided by the workers at the beginning and could be further refined from their historical selected tasks. To ensure the task completion quality, each work can only undertake a limited number $n_{\max}$ of tasks concurrently. However, to provide each worker with more choices, when the system assigns tasks, it will not just display the top $n_{\max}$ tasks, instead, it will display more, e.g.,

TABLE II
MOTIVATION EXAMPLE: 20 TASKS OF FOUR CATEGORIES ARE ASSIGNED TO FOUR WORKERS. NUMBERS IN PARENTHESES IN THE THIRD COLUMN INDICATE PREFERENCE SCORES OF WORKERS FOR EACH TASK CATEGORY. THE HIGHER THE SCORE IS, THE MORE FAVORITE BY THE WORKER. FOR SIMPLICITY, WE SET BOTH n_{cand} AND $n_{\max}$ AS 5 IN THIS EXAMPLE

Worker	Attribute preference weight	Preference score	Task list via traditional method	Task list with better task coverage
w_1	$< 0.2, 0.1, 0.2, 0.2, 0.3 >$	$c_4(10), c_2(9), c_1(8), c_3(1)$	$t_1^{c_4}, t_2^{c_4}, t_3^{c_4}, t_4^{c_4}, t_5^{c_4}$	$t_1^{c_1}, t_2^{c_1}, t_3^{c_1}, t_4^{c_1}, t_5^{c_1}$
w_2	$< 0.2, 0.2, 0.1, 0.1, 0.4 >$	$c_4(10), c_3(9), c_2(2), c_1(1)$	$t_1^{c_4}, t_2^{c_4}, t_3^{c_4}, t_4^{c_4}, t_5^{c_4}$	$t_1^{c_4}, t_2^{c_4}, t_3^{c_4}, t_4^{c_4}, t_5^{c_4}$
w_3	$< 0.1, 0.1, 0.3, 0.3, 0.2 >$	$c_3(10), c_4(3), c_1(2), c_2(1)$	$t_1^{c_3}, t_2^{c_3}, t_3^{c_3}, t_4^{c_3}, t_5^{c_3}$	$t_1^{c_3}, t_2^{c_3}, t_3^{c_3}, t_4^{c_3}, t_5^{c_3}$
w_4	$< 0.2, 0.1, 0.1, 0.3, 0.3 >$	$c_4(10), c_2(9), c_1(2), c_3(1)$	$t_1^{c_4}, t_2^{c_4}, t_3^{c_4}, t_4^{c_4}, t_5^{c_4}$	$t_1^{c_2}, t_2^{c_2}, t_3^{c_2}, t_4^{c_2}, t_5^{c_2}$

top n_{cand}, candidate tasks (after the preference matching) for the worker to pick. After the worker finishes each task $t_i^{c_j}$, she will receive $r_i^{c_j}$ reward as the return.

Task Assignment Problem: With the notations above, prior task assignment designs could convert the matching between task attributes and worker preferences into a score $\mathcal{S}_{\text{match}}$ and conduct the assignments by the following optimization:

$$\max\ \mathcal{S}_{\text{match}}$$

with certain constraints, like each task is undertaken by a certain number of workers at most. Note that the calculated matching score $\mathcal{S}_{\text{match}}$ can be purely based on the preference matching or weighted by some other performance indicators [35], e.g., latency, task completion quality, etc.

As the assignment is essentially a score matching problem, solving it directly will lead to a *misaligned task coverage problem*, i.e., some popular tasks tend to enter workers' candidate task lists (with higher matching scores), while some less popular tasks could be unsuccessfully assigned. As the task reward $r_i^{c_j}$ is usually one task attribute, to encourage these tasks being undertaken on time, the system may have to increase their biding costs to reassign the tasks, which essentially increases the overall costs of crowdsensing systems. Next, we detail this issue through one concrete example.

B. Motivation Example

In this example, we consider the four task categories for the urban traffic monitoring with multisource data collections.

1) *Category c_1:* Measuring the vehicle traffic flow at a specific intersection, denoted as $\{t_1^{c_1}, t_2^{c_1}, \ldots\}$.
2) *Category c_2:* Measuring the passenger flow at a specific bus stop, denoted as $\{t_1^{c_2}, t_2^{c_2}, \ldots\}$.
3) *Category c_3:* Checking whether there is any traffic accident or construction at a specific region, denoted as $\{t_1^{c_3}, t_2^{c_3}, \ldots\}$.
4) *Category c_4:* Checking the surface condition of a specific road, denoted as $\{t_1^{c_4}, t_2^{c_4}, \ldots\}$.

For simplicity, these categories share the same attribute vector $\vec{\mathbf{a}}$ with the following five attributes.

1) *a_1:* Traveling distance for completing the task (in km).
2) *a_2:* The latest beginning time from now (in hour).
3) *a_3:* Duration for completing the task (in hour).
4) *a_4:* Type of the desired sensing data.
5) *a_5:* Reward for completing a task. In this example, we adopt a simple reward formula, e.g., $r_i^{c_j} = (a_1 + 2 * a_3) * \alpha$, where α (≥ 1) is an augment factor when more rewards are needed to attract workers, to illustrate the design.

In this example, we assume the attribute vector instances of tasks of the four categories are <1, 1, 0.5, numeric, $2 * \alpha$>, <0, 1, 1, numeric, $2 * \alpha$>, <2, 0, 1, numeric, $4 * \alpha$>, and <1.5, 3, 1, image, $3.5 * \alpha$>, respectively. Assuming we have five tasks for each category, and thus, there are totally 20 tasks for assignments. Meanwhile, four workers are available and each can undertake five tasks at most, i.e., $n_{\max} = 5$.

Given an instance of each worker's preferences to each attribute, as illustrated in Table II, existing approaches [15], [25] first derive the task preference scores for each worker, based on which the task assignment can be conducted. Table II shows that *Category c_4* is the most popular task among workers and its tasks are recommended to w_1, w_2, and w_4 at the same time. Although traditional methods can guarantee a high matching score (since it assigns the most preferred tasks to each worker), we find that some tasks could be never recommended at all, i.e., tasks of *Categories c_1* and c_2. As a result, it is expected that more rewards are needed to attract workers to accept those tasks, e.g., doubling their rewards by changing α from 1 to 2. Therefore, the traditional methods can only achieve 50% of task coverage and costs will be increased for the coverage. In this case, the overall cost needs to increase to 77.5 (we omit the details due to page limit), so that all tasks can be covered.

However, for those unassigned tasks, purely from the feature's matching point of view, there already exist sufficient workers who likely accomplish them. In other words, these tasks can be potentially assigned. In this example, we demonstrate an alternative way to accomplish the task assignment, as shown in the last column of Table II, which may achieve a better tradeoff between the task coverage and the matched scores. In particular, we can assign tasks of *Category c_4* to w_2, tasks of *Category c_2* to w_4, and tasks of *Category c_1* to w_1. Such an assignment achieves 100% coverage with overall cost only 57.5, while the matched score is still as high as 185 (compared with the original score 200).

C. Design Overview of cTaskMat

Inspired by this example, we propose the *cTaskMat* design in this article. The key idea behind is that we migrate certain workers from their most popular tasks to the less popular ones (but still with sufficiently high preference scores), so that the overall task coverage can be increased and thus the potential cost can be saved. Fig. 1 illustrates the system architecture with three major components: 1) *worker profiling*; 2) *coverage-ensured task assignment*; and 3) *cost-efficient remedy*.

First, the *worker profiling* module implicitly learns each worker w_i's task preferences from historical records and thus

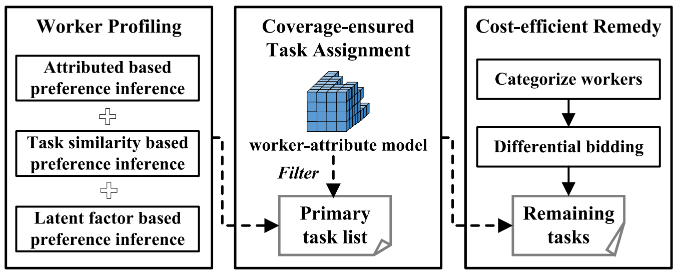
Worker Profiling
Attributed based preference inference
Task similarity based preference inference
Latent factor based preference inference
Coverage-ensured Task Assignment
worker-attribute model
Filter
Primary task list
Cost-efficient Remedy
Categorize workers
Differential bidding
Remaining tasks

Fig. 1. Illustration of the system architecture of *cTaskMat*.

produces a primary task list ordered by w_i's task preference scores. Then, the *coverage-ensured task assignment* module further infers w_i's attitudes on task attributes and filters out these tasks containing attributes disliked by w_i to derive top n_{cand} candidate tasks for w_i. Finally, to assign remaining tasks, the *cost-efficient remedy* module classifies workers and differentially raises task rewards according to each worker type. The final outputs of *cTaskMat* are the assigned task lists for all workers.

III. Design of *cTaskMat*

In this section, we elaborate the design detail of each component of *cTaskMat* in the previous three sections, respectively, and present the discussions in the last section.

A. Worker Profiling

We characterize each worker w_i's preferences on tasks and then make use of such information to generate a primary task list for w_i. To this end, we exploit w_i's historical task selection information in an implicit manner, due to the lack of worker's explicit feedback (e.g., rating, like, or dislike). Specifically, we infer workers' task preferences by comprehensively considering the task attributes, task similarity, and latent factor between workers and tasks.

1) Attributes-Based Preference Inference: Each task is described by an attribute vector $\vec{\mathbf{a}}$,[1] which covers the aspects of traveling distance, duration, reward, etc. For a given task t_j recommended by the system, a worker's decision (i.e., select or not) can be viewed as the result through implicitly scoring on the attributes of t_j. Therefore, we can regard each worker's historical task selection records, including the decisions and corresponding task attributes, as the training samples to build some machine learning models to capture each worker's attribute-based preferences. For a given task t_j, we take its attribute vector $\vec{\mathbf{a}}$ as the input of the learned model for worker w_i to derive a probability p_{ij}, which indicates how likely w_i will select task t_j. Indeed, there exist multiple candidate models, e.g., k-means, SVM, logistic regression [11], or Bayesian classifier [23], for this learning task. Previous studies [2], [9], [24] find that the Bayesian classifier model is less likely to be overfitting and can achieve a better generalization. Hence, we adopt the Bayesian classifier in our implementation.

[1]We omit the superscript of category when there is no confusion.

2) Task Similarity-Based Preference Inference: In addition to the detailed attributes of each individual task, we could infer the worker's task preferences by exploiting the similarity between tasks. The intuition behind is that tasks similar with already completed tasks by worker w_i may be preferred by w_i in the future. Therefore, item-based collaborative filtering [25] can be used to predict w_i's preferences on those tasks. Here, tasks are viewed as items. For any two tasks, t_j and t_k, we calculate their similarity θ_{jk} using the following equation:

$$\theta_{jk} = \frac{|W_j \cap W_k|}{|W_j \cup W_k|} \tag{1}$$

where W_j and W_k represent the worker sets having selected task t_j and task t_k in the past, respectively. Different from correlation or cosine-based similarity functions used in the previous works [25], (1) measures the task similarity from the perspective of workers' practical choices. It is worthy to note that item-based collaborative filtering allows us to compute the similarity of any pair of tasks in advance, which can speedup the online task assignments.

Denote $\mathbf{U}$ as the worker–task preference matrix, where each entry u_{ij} is worker w_i's preference on task t_j. Based on the similarity function in (1), we calculate u_{ij} using

$$u_{ij} = \sum_{k \in \left(D_{w_i} \cap S_j^e\right)} \theta_{jk} \tag{2}$$

where D_{w_i} is the task set already completed by w_i, and S_j^e is the top-e most similar tasks with t_j.

3) Latent Factor-Based Preference Inference: Compared to the previous two inference methods, we further use the latent factor model (LFM) to indirectly infer worker's interests on tasks. In recommender systems, LFM is widely used to group items into virtual clusters (i.e., latent factors) and explore the relations between clusters and users/items for recommendations. In our case, we can also leverage latent factors to link the worker's preferences and tasks. Similarly, we construct a worker–task preference matrix, whose elements l_{ij}, indicating worker w_i's preference on task t_j, is set as

$$l_{ij} = \begin{cases} N/A, & \text{if } w_i \text{ did not browse } t_j \\ -1, & \text{if } w_i \text{ browsed but not selected } t_j \\ 1, & \text{if } w_i \text{ browsed and selected } t_j. \end{cases}$$

The LFM aims to infer the missing elements in the worker–task preference matrix through matrix factorization, which decomposes the worker–task preference matrix into two matrices, P and Q. We can infer the missing workers' task preferences by minimizing the following loss function:

$$\begin{aligned} L &= \sum_{(i,j)\in K} \left(l_{ij} - \hat{l}_{ij}\right)^2 \\ &= \sum_{(i,j)\in K} \left(\left(l_{ij} - \sum_{k=1}^{K}(p_{ik} \times q_{jk})\right)^2 + \lambda\|\vec{p}_i\|^2 + \lambda\|\vec{q}_j\|^2\right) \end{aligned}$$

where $\hat{l}_{ij}$ is the predicted value by LFM, K is the specific number of latent factors, $\vec{p}_i$ is the ith row of P that captures the relations between worker w_i and latent factors, and q_j is the jth column of Q that captures relations between task t_j and latent factors. λ is a regularization parameter to avoid overfitting. We set $\lambda = 5$ in our implementation.

Comprehensive Task Preference Score: The above three inference results estimate worker's task preference from three different aspects, which allow us to compound a more comprehensive task preference score. Specifically, we combine these inferences to derive w_i's preference score on task t_j (denoted as s_{ij}) using a weighted linear model, i.e.,

$$s_{ij} = \mu_1 \times p_{ij} + \mu_2 \times u_{ij} + (1 - \mu_1 - \mu_2) \times \hat{l}_{ij} \tag{3}$$

where μ_1 and μ_2 are both user-defined parameters. Based on our empirical testing, we set $\mu_1 = 0.24$ and $\mu_2 = 0.38$ in our current implementation.

By calculating the preference scores of all tasks with respect to worker w_i, *cTaskMat* sorts these tasks in the descending order based on their preference scores and regards the sorted list as w_i's primary task list.

B. Coverage-Ensured Task Assignment

Traditional task assignment methods usually adopt the *task acceptance rate* as an important metric to evaluate the performance of crowdsensing systems [14], which is defined as

$$\text{Accep\_rate} = \frac{1}{|\mathcal{W}|} \sum_{w_i \in \mathcal{W}} \frac{|R_{w_i} \cap T_{w_i}|}{|R_{w_i}|} \tag{4}$$

where R_{w_i} and T_{w_i} are the assigned task set and truly selected task set, respectively, for worker w_i.

As motivated in Section II-B, although traditional methods can achieve the high acceptance rate, they fail to guarantee task coverage, which may introduce extra costs for assigning the less popular tasks. Formally, we define *task coverage rate* as

$$\text{Cover\_rate} = \frac{\left|\bigcup_{w_i \in \mathcal{W}} R_{w_i}\right|}{|\mathcal{T}|}. \tag{5}$$

According to the above definitions, we find the assigned task list R_{w_i} is the key that will decide both the task acceptance rate and task coverage rate. Typically, a larger set R_{w_i} can benefit the improvement of Cover_rate. However, for a crowdsensing system, the number of assigned tasks, i.e., n_{cand}, is fixed. As a result, we can only adjust the tasks in each R_{w_i} to optimize the performances of both the task acceptance rate and task coverage rate, which is formally stated as

$$\max\{\text{Accep\_rate}, \ \text{Cover\_rate}\}.$$

To achieve this goal, we will explicitly mine workers' preferences on task attributes and exploit such information to process the primary task list to derive the suitable n_{cand} tasks for each worker. Specifically, we make use of worker's favorite task attributes (i.e., positive features) to enhance the task acceptance rate, and meanwhile, exploit worker's dislike task attributes (i.e., negative features) to filter out tasks owning the disliked attributes from a primary task list. The latter operation will increase the recommendation chances for the less popular tasks, and thus potentially improve the task coverage rate.

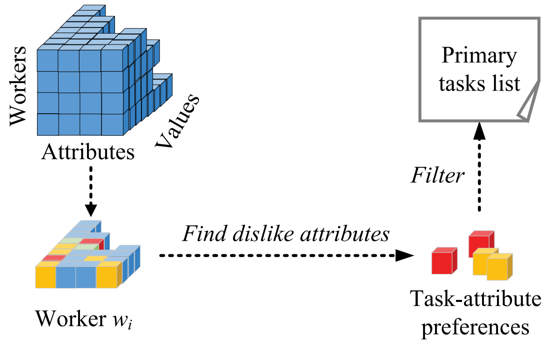

Fig. 2. Illustration of the worker-attribute model $\mathbb{A}$.

Empirical Survey Result: We have conducted a questionnaire survey (see more details in Section IV-A), and one of the questions is "*Will you directly reject one task when it owns some task attributes you do not like?*" The statistical result indicates that most respondents (around 82%) choose *Yes* and will directly reject the task. Such a result implies that it is feasible and necessary to mine workers' preferences on task attributes and adjust the assigned task list accordingly.

Worker-Attribute Model: To capture workers' preferences on task attributes, we construct a tensor $\mathbb{A} \in R^{|\mathcal{W}| \times m \times v}$ with the three dimensions standing for workers of size $|\mathcal{W}|$, task attributes of size m, and attribute values of size v, respectively. An entry $\mathbb{A}_{(i,j,k)}$ denotes the preference of worker w_i on the kth possible value of the jth task attribute. The top left of Fig. 2 demonstrates a worker-attribute model, and the bottom left of this figure shows an instance of task attribute preferences for worker w_i. The color of a box (i.e., an entry) in Fig. 2 indicates worker w_i's attitude on the given value of a specific task attribute. In this example, the redder a box is the more disliked by the worker. Next, we will introduce how to initialize and update this model.

Initialization: To initialize the model $\mathbb{A}$, a common approach is to assign the entries with random weights, while we aim to boost the model initialization by exploiting previously derived results. To this end, we reduce the 3-D tensor $\mathbb{A}$ to a 2-D matrix $\mathbf{A} \in R^{|\mathcal{W}| \times (mv)}$ by stacking all attribute values of all attributes sequentially. Then, we can reuse the LFM technique in Section III-A3 to decompose the worker–task preference matrix and derive two matrices P and Q. In principle, we can set the size of matrix P as $|\mathcal{W}| \times (mv)$. Recall that P captures the relations between workers and latent factors, and we thus could initialize $\mathbf{A}$ as matrix P by implicitly viewing each latent factor in P as a specific task attribute value. By mapping the items in $\mathbf{A}$ back to the entries of $\mathbb{A}$, we obtain the initial worker-attribute model.

Since matrix P has been calculated in the prior step, it can be reused directly for the initialization without introducing additional computation overhead. More importantly, the items in matrix P are meaningful in capturing the relations between workers and task attributes. Therefore, compared with the random initialization, our initialization can greatly accelerate the convergence of the worker-attribute model updating.

Update: We update the worker-attribute model $\mathbb{A}$ continuously with the latest task assignment results. Since there is no direct feedback from workers on task attributes and attribute values, e.g., ratings, we can only implicitly learn the preferences through workers' historical behavior on task selections. Fortunately, a crowdsensing system could log workers' behaviors. Such records include which tasks the worker had browsed, selected, and finally completed. Note that workers can only browse the tasks recommended by the system. We regard the tasks in records as training samples and classify them into two groups.

1) *Positive Samples:* When task t_j was browsed, selected, and completed by worker w_i.
2) *Negative Samples:* When task t_j was browsed but not selected by worker w_i.

We use the accumulated positive and negative samples to update the worker-attribute model $\mathbb{A}$ by exploiting the idea of the attention mechanism. Recently, the attention mechanism has been successfully applied in many domains, e.g., recommender system [4], information retrieval [39], and computer vision [5], and has demonstrated remarkable performance improvements. The key idea of the attention mechanism is to distribute different weights to each part of a vector of interest based on a similarity function that can predict the attention score [29]. Here, we make use of the attention mechanism to explore the influences of different attributes and their values on the acceptance or rejection of a task for a given worker. Specifically, we use the following similarity function to calculate an attention score vector when assigning task t_j to worker w_i

$$\mathbf{s_{ij}} = \mathbf{w_i} \odot \mathbf{t_j} \oplus \left|\min\left(\mathbf{w_i} \odot \mathbf{t_j}\right)\right. \tag{6}$$

where $\mathbf{w_i}$ is a worker-attribute vector that describes the worker's interest on each task attribute, and $\mathbf{t_j}$ is a vector that describes the relations between this task and attribute values. In addition, the operator $\odot$ denotes the elementwise product of two vectors, function min() returns the minimum value of a given vector, and the operator $\oplus$ denotes to add the right value to each element of the left vector.

The result $\mathbf{s_{ij}}$ of (6) indicates the attention scores of the worker with respect to the attribute values of the given task. Each item in vector $\mathbf{s_{ij}}$ implies to what degree the worker will prefer the corresponding attribute value. Such information can be used to update the worker-attribute model. In our design, rather than directly applying $\mathbf{t_j}$ to update the worker-attribute model, we further utilize (7) to calculate an updated weight vector. For each attention score $s_z \in \mathbf{s_{ij}}$, its corresponding weight $g_z \in \mathbf{g}_{ij}$ is computed by the following equation:

$$g_z = \frac{\exp(s_z)}{\sum_{s \in \mathbf{s_{ij}}} \exp(s)}. \tag{7}$$

Based on the weight score vector $\mathbf{g}_{ij}$, we then update $\mathbf{w_i}$ of the worker-attribute model given the task t_j is a positive sample or a negative sample, using the following equation:

$$\mathbf{w_i} = \mathbf{w_i} + (-1)^I \mathbf{g_{ij}} \odot \mathbf{t_j} \tag{8}$$

where $I = 0$ if the task t_j is a positive sample and $I = 1$ if task t_j is a negative sample. Therefore, we exploit an attention-based mechanism to explore the worker's preferences on task attributes and attribute values, and update the worker-attribute model pertinently.

In practice, we can train the worker-attribute model $\mathbb{A}$ with historical records, and this model can be continuously updated when more worker behaviors are logged. Therefore, even a worker changes her preferences in the future, model $\mathbb{A}$ will also be updated according to her latest task selection choices.

Process Primary Task List Using $\mathbb{A}$: The derived model $\mathbb{A}$ can be used to process each worker w_i's primary task list by filtering out the tasks containing w_i's obviously dislike attributes and attribute values. In theory, when an entry $\mathbb{A}_{(i,j,k)}$ has a negative value, it means that worker w_i does not like this task since w_i has a negative preference on its jth attribute with the kth value. In practice, we adopt a more strict rule by setting a threshold η below 0 (e.g., -0.5 in our implementation). When $\mathbb{A}_{(i,j,k)} < \eta$, *cTaskMat* considers that worker w_i dislikes the task which has an attribute a_j with value equaling to v_k. For each task in the primary task list, *cTaskMat* checks its attribute values by comparing with the worker-attribute preference in $\mathbb{A}$. *cTaskMat* simply filters out the task once it has a low attribute preference for the worker.

By doing so, *cTaskMat* adjusts the primary task list and picks the top n_{cand} (or all when there are not enough) tasks in the list as the assigned task list for each worker.

C. Cost-Efficient Remedy

After worker-attribute model-based task assignment in the previous section, there may be still some remaining tasks that are not assigned yet. The traditional methods usually raise the rewards of the remaining tasks to attract potential workers. Such a method, however, is not cost efficient and may not guarantee the system performance, e.g., task acceptance rate, when tasks are not recommended to the right workers.

Therefore, we propose a cost-efficient remedy method to reassign the remaining tasks to these available workers, whose candidate task lists are not full (with the number of already assigned tasks $< n_{\text{cand}}$). Specifically, our method exploits the worker-attribute model $\mathbb{A}$ to categorize the workers and uses different strategies to reassign the remaining tasks to the workers of each group. We first compute an average worker-attribute profile $\mathcal{A}^{\text{ave}}$ by averaging the preferences of each attribute value for all workers, i.e., $\mathcal{A}^{\text{ave}}_{(j,k)} = \text{mean}(\mathbb{A}_{(*,j,k)})$. From model $\mathbb{A}$, we can also extract worker w_i's attribute preference profile $\mathcal{A}^i$. By comparing $\mathcal{A}^i$ with $\mathcal{A}^{\text{ave}}$, we can derive the preference difference $\mathcal{A}^{i/\text{ave}}$ of worker w_i against the average attribute preferences of all workers, and classify w_i as follows.

1) *Rigorous worker*, when there are at least n elements in $\mathcal{A}^{i/\text{ave}}$ larger than a threshold γ (we empirically set $n = 20$ and $\gamma = 0.5$ in our implementation). Rigorous workers have strong preferences and significantly dislike some task attributes.
2) Otherwise, *nonrigorous worker*, who have relatively broad interests in all task attributes. We further classify

nonrigorous workers according to their attitudes on the task rewards. Specifically, for a nonrigorous worker w_i, we compare her attribute preference profile $\mathcal{A}^i$ with $\mathcal{A}^{\text{ave}}$ only on the attribute of task reward, and regard w_i as follows.

a) *Lucrative worker*, who prefers high task rewards, e.g., task rewards of 80% prior completed tasks are higher than the average task reward computed from $\mathcal{A}^{\text{ave}}$.

b) *Normal worker*, who does not care task rewards too much and the rewards of prior completed tasks are uniformly distributed.

By categorizing all workers according to their attitudes on task attributes, we can differentially recommend tasks to them while guaranteeing the task acceptance rate and saving the cost. Specifically, we do not reassign a task t_j to a rigorous worker w_i when t_j contains attributes strongly disliked by w_i, and we only largely increase the reward of task t_j when it is recommended to a lucrative worker. For each remaining task t_j, it is reassigned to workers as the following orders and reward raising strategies.

1) t_j is first reassigned to rigorous workers if possible. For a rigorous worker w_r, if t_j does not contain attribute values disliked by w_r, it is recommended to w_r. The task reward is not raised.
2) Otherwise, t_j is reassigned to possible normal workers, and its reward is slightly raised, e.g., increasing by 25%.
3) Finally, t_j is recommended to lucrative workers, and the task reward is largely raised, e.g., increasing by 50%.

Note that each worker can only be assigned n_{cand} tasks. With the remedy assignments, *cTaskMat* can finally assign all the tasks to suitable workers with few increases on the system cost. In Section IV-B, our experimental result proves that *cTaskMat* can guarantee system performances while significantly reducing the system costs. Algorithm 1 summarizes the task assignment algorithm of *cTaskMat*.

Algorithm 1: Task Assignment Algorithm of *cTaskMat*

```
Input: worker set $\mathcal{W}$, tasks set $\mathcal{T}$, historical records
Output: assigned task list for each worker
1  foreach worker $w_i$ do
2      Retrieve $w_i$'s historical records;
3      foreach task $t_j$ in historical records do
4          Calculate $s_{ij}$ using Eq. (3);
5      end
6      Creat primary task list for $w_i$ by sorting all tasks
       according to their $s_{ij}$ in descending order;
7  end
8  Construct the worker-attribute model $\mathbb{A}$;
9  foreach worker $w_i$ do
10     foreach attribute $a_j$ with k-th value $v_k$ do
11         if $\mathbb{A}_{(i,j,k)} < \eta$ then
12             Filter out tasks having attribute $a_j$ and its
               value equaling to $v_k$ from $w_i$'s primary task
               list;
13         end
14         Pick the top $n_{cand}$ tasks to form assigned task set
           for $w_i$;
15     end
16 end
17 Categorize worker $w_i$ based on $\mathcal{A}^i$ and $\mathcal{A}^{ave}$;
18 Reassign remaining tasks to rigorous, normal, and
   lucrative workers sequentially;
19 return assigned task list for each worker $w_i$;
```

D. Discussions

Handling the Cold Start Issue: When *cTaskMat* is initially adopted, there are no historical records. As a result, we cannot learn workers' task preferences and build the worker-attribute model from workers' behaviors. To cope with such a common *cold start problem*, we can conduct a questionnaire survey for each new worker to rate a series of preference attributes and thus derive some preliminary information about her general preferences. With this profile as a seed, our system is able to assign tasks already, guided by the worker's preferences.

In addition to the survey-based worker profiling, *cTaskMat* further enhances the worker's preferences based on the procedure of task recommendations and selections. Specifically, *cTaskMat* recommends the worker a list of candidate tasks, which cover the aspects of tasks that overlap with the worker's profile. On the other hand, the worker selects these tasks that she is willing to complete. Such manually selected tasks can better reflect the worker's true preferences, and the task selection results can be exploited by *cTaskMat* to appropriately update the worker-attribute model $\mathbb{A}$.

Handling the Worker's Dynamic Preferences: Given the task selection history of a worker, the system always aims to capture the (current) preferences of this worker. It is indeed possible that these preferences may not reflect the actual situation or they may vary gradually over time. Therefore, *cTaskMat* allows a worker to proactively launch the profile updating, e.g., amending the rating for each preference attribute in the survey. Based on the updated preferences, the candidate task list could contain the tasks, which were disliked by the worker before but are acceptable by this worker now and in the future. In addition, the worker will also be asked periodically about whether her preferences need to be updated if no update is received so far. By doing so, the task assignment can follow the preferences varying better.

Impact of Filtering Primary Task List: cTaskMat makes use of model $\mathbb{A}$ to filter out some tasks from each worker w_i's primary task list, which thus increases the opportunity of recommending less popular tasks to w_i. For those filtered tasks, they still may be assigned to other workers or reassigned with the cost-efficient remedy at last. Therefore, the filtering process will not affect the task completions.

Settings of System Parameters: cTaskMat involves the settings of some system parameters to measure workers' preferences on tasks and attributes. In general, these parameters can be empirically set according to the system's requirements. Since we adopt the same parameters to evaluate each worker and task, their impact on system performance can be ignored.

IV. Evaluation

In this section, we implement *cTaskMat* and extensively evaluate its performances by comparing with baseline methods on the real-world experiment and data-driven evaluations.

A. Experiment Setup

We implement *cTaskMat* in Python and run the system in a powerful Linux server that has 3.6-GHz CPU with 8 Core and 8-GB memory. We use the data collected from a real-world questionnaire survey and MovieLens 100K data set [8] to evaluate and compare the performances of *cTaskMat* and baseline methods on some widely used performance metrics. By default, we initially set $\alpha = 1$ in the task reward formula and increase it if necessary to reassign the remaining tasks. Besides, we set $n_{\text{cand}} = 20$ and $n_{\max} = 15$.

Questionnaire Survey: We conduct a questionnaire survey and recruit 74 volunteers (47 males and 27 females) to participate in the experiment. Specifically, we manually create five task categories for urban traffic monitoring. Besides the four task categories mentioned in Section II-B, we add another task category as *Sharing your latest traveling trajectory.* We instantiate 140 tasks for each category and thus totally have 700 tasks for the 74 volunteers. All tasks share the same attribute vector $\vec{\mathbf{a}} = <a_1, a_2, a_3, a_4, a_5>$ just as mentioned in Section II-B, while each task will have a different settings of the attribute values. The entire task corpus is shown to each volunteer w_i through the questionnaire survey, and w_i will decide whether to accept a task based on his/her preferences on the task attributes. If a task is not selected by volunteer w_i, the survey allows w_i to write down an anticipate reward that can persuade him/her to accept the task. Such a setting helps us to analyze workers' attitudes on task reward and understand how to raise task rewards to attract workers. For the data collected from this questionnaire survey, we use 80% of all data to learn workers' task preferences and build the worker-attribute model $\mathbb{A}$, and exploit the remaining 20% for testing.

MovieLens 100K Data Set: In addition to our questionnaire survey data set, we choose the publicly available MovieLens data set [8] to examine the performances of our system in the large-scale application scenario. In this data set, each movie (similar as "task" in our system) is associated with certain attributes and users select movies according to their preferences on these attributes. Therefore, the settings of movie scoring are similar as that in our system. Specifically, this data set contains 100 000 ratings (1–5, the higher the better) from 943 users on 1682 movies, and each user had rated at least 20 movies. We treat the 4-star and 5-star ratings as the positive feedback and treat other ratings as the unknown feedback. Similarly, we split the data set into training data and testing data according to the 80%/20% rule. In the experiments, we assume each worker can score a movie through her smartphone and we would like to rate a number of movies by crowdsourcing to the workers.

Baseline Methods: We compare *cTaskMat* with the following alternative task assignment methods.

1) *Random Assignment (RA):* The method will randomly assign n_{cand} tasks for each worker.

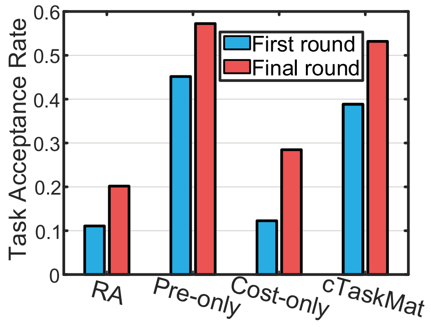

Fig. 3. Performance comparisons of different methods on the task acceptance rate.

2) *Preference-Only Task Assignment (Preonly)*: This method assigns tasks to each worker based on their preferences merely, which is implemented based on the classical item-based collaborative filtering technique [25].
3) *Cost-Only Task Assignment (Cost-Only):* Linden *et al.* [15] proposed a cost-efficient collaborative filtering method, and we implement it for task assignments.
4) *IRGAN [30]:* It is one of the state-of-the-art methods for item-based recommendation and can also be applied for task assignments. Since it needs a lot of data to train the generative adversarial nets (GANs), we only compare it on the large MovieLens 100K data set.

Performance Metrics: We compare *cTaskMat* and baseline methods on the task acceptance rate [i.e., Accep_rate defined in (4)] and task coverage rate [i.e., Cover_rate defined in (5)]. Besides, we define the *Cost* metric to evaluate the system cost for completing all tasks, which is calculated as

$$\text{Cost} = \sum_{w_i \in \mathcal{W}} \sum_{t_j \in T_{w_i}} r_j^c \tag{9}$$

where T_{w_i} means the selected task set of worker w_i and r_j^c is the reward for completing task t_j of category c.

B. Results of the Real-World Experiment

To assign tasks to suitable workers, *cTaskMat* mainly takes two steps.

1) It first makes use of each worker w_i's task preferences to derive a primary task list and then filters out these tasks, which contain dislike attributes by exploiting the worker-attribute model, from the list to obtain the top n_{cand} assigned tasks (denoted as the *core* step).
2) It then explores workers' attitudes on task rewards to intentionally raise the reward of each unassigned task so that to minimize the overall system cost (denoted as the *remedy* step). We evaluate impacts of the *core* step and *remedy* step on *cTaskMat*'s performances.

Impacts on Accep_rate and Cover_rate: We run baseline methods, *cTaskMat* and *cTaskMat-core* (excluding the *remedy* step from the full version) on the testing data set and present their performance results of Accep_rate and Cover_rate in Figs. 3 and 4. Each method usually runs several rounds to assign all tasks and may raise task rewards in the following

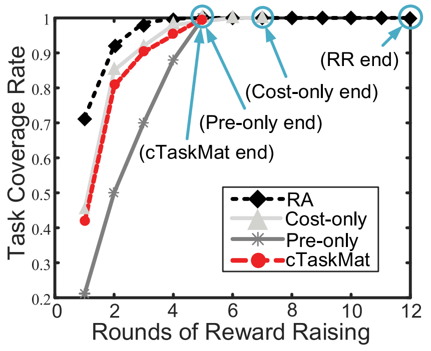

Fig. 4. Performance comparisons of different methods on the task coverage rate.

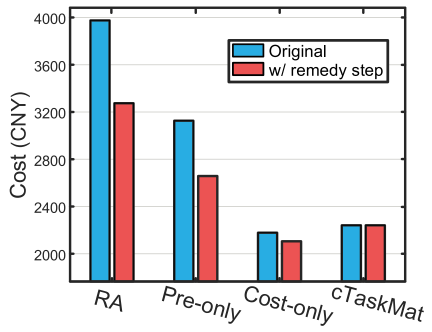

Fig. 5. Overall costs (in CNY) of different methods and the effectiveness of the *remedy* step.

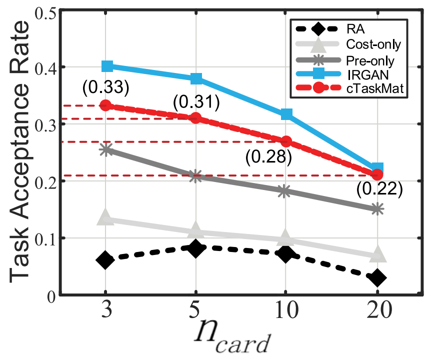

Fig. 6. Task acceptance rate on a large data set.

rounds. We report the task acceptance rates of the first round and final round for all methods in Fig. 3. We can see the acceptance rates of all methods in the first round are low, and will increase at the last round. Among all the methods, the *Preonly* method achieves the highest task acceptance rate (about 0.57 in the final round) since it only recommends the most preferred tasks to each worker, while *RA* and *Cost-only* have the worst performances, with the final Accep_rate less than 0.30. The task acceptance rate of *cTaskMat-core* is 0.39 and increase to 0.53 with the help of the *remedy* step, with 36% improvement on Accep_rate.

Fig. 4 shows the task coverage rates for different methods along with rounds of reward raising. In essence, to reassign the remaining tasks, more and more task rewards are needed at last. *Preonly* and *cTaskMat* achieve full task coverage with the fewest rounds, while *Preonly* has the worst performance on Cover_rate in the first round. As a result, *Preonly* has to raise rewards for a lot of remaining tasks in the next rounds. Although *Cost-only* shows similar performance as *cTaskMat* in the initial stage, its low Accep_rate causes more rounds (and thus more system cost) are needed. It is no surprise that *RA* achieves full task coverage with the most rounds. Since *RA* blindly assigns tasks to workers, which results in a low acceptance rate and more attempts are needed.

From the results in Figs. 3 and 4, we can see that among all the methods *cTaskMat* achieves the best performance with both the high task acceptance rate and task coverage rate at the same time. Specifically, the *core* step helps *cTaskMat* assign workers with their most preferred tasks, and the *remedy* step reassigns the remaining tasks to achieve full task coverage in a cost-efficient manner. In the next, we will study how the *remedy* step helps *cTaskMat* save costs during the assignment of less popular tasks.

Impacts on Cost: To assign the remaining tasks to workers, a widely adopted strategy is to raise task rewards for attracting workers and such a raising increases along with the rounds. In each round, the previous methods equally raise the rewards of tasks for all available workers (e.g., raising by 50%), while *cTaskMat* first analyzes workers' attitudes on task attributes and rewards, and then differentially raise task rewards for different workers. We report the task reassignment cost comparisons for all methods in Fig. 5, where blue bars represent the costs for original methods and red bars represent the costs if we apply our *remedy* step in those methods for task reassignments. From the experimental results, we can see the *Cost-only* method introduces the fewest costs while *Preonly* has the most costs to reassign the less popular tasks. *cTaskMat* has a moderate cost of about 2241. When we apply the *remedy* step to those methods, we can see obvious reductions on their costs, saving 32.3%, 35.1%, and 12.3% costs compared to the original strategy for *RA*, *Preonly*, and *Cost-only*, respectively. Although the *Cost-only* method is designed to save costs for task assignments, our *remedy* step can further reduce its costs.

The *core* step of *cTaskMat* will filter out some tasks from each worker's primary task list, and we have tracked how these tasks are reassigned. According to our experiment, we find that although these tasks are removed from a worker's primary task list, 74.0% of them will be successfully reassigned to other workers. For the remaining tasks, the *remedy* step reassigns them to suitable workers by carefully setting task rewards. Thanks to the analysis of workers' attitudes on rewards, 20.5% tasks are reassigned in the *remedy* step without raising rewards and only 5.5% tasks are reassigned after bidding.

These results from the real-world experiment demonstrate that *cTaskMat* can well balance various performance metrics than the alternatives. Specifically, *cTaskMat* could achieve high task coverage and largely save the system cost on average by 28% while sacrificing task acceptance by 7% when compared to *Preonly* and *Cost-only*, the state-of-the-art methods.

C. Results of Data-Driven Evaluations

We make use of the publicly available MovieLens data set to examine *cTaskMat*'s performances in a large-scale application

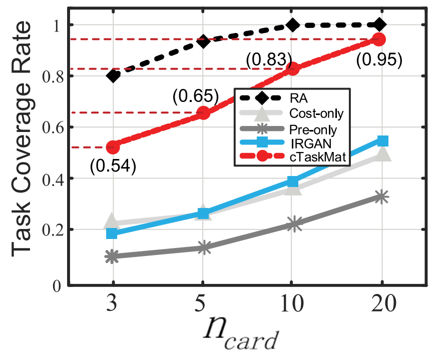

Fig. 7. Task coverage rate on a large data set.

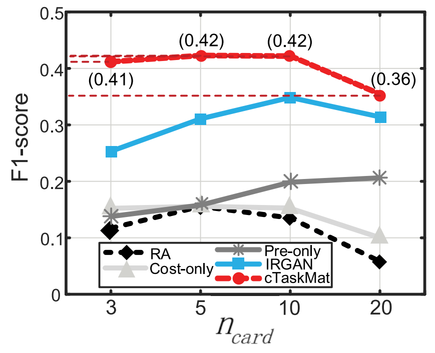

Fig. 8. F1-score on a large data set.

scenario. Figs. 6 and 7 present the performance comparisons of different methods when we vary the number of maximum assigned tasks, i.e., n_{cand}, from 3 to 20. Specifically, when we increase n_{cand}, task acceptance rates of all methods are reduced, as shown in Fig. 6. This is because each worker can undertake a limited number of tasks, and thus among the candidate task set, only a fixed number of tasks are finally selected. Among all the methods, *cTaskMat* takes the second place and has comparable performance as IRGAN, one of the state-of-the-art methods. With respect to the task coverage rate of the first round as shown in Fig. 7, our method performs much better than IRGAN. Although *RA* can achieve the high task coverage rate, it still cannot guarantee a good acceptance rate due to its blind assignments.

In order to comprehensively compare all methods, we calculate their F1-score in different n_{cand} settings and report the results in Fig. 8. When calculating F1-score, we replace recall as the task coverage rate and use F1-score as a comprehensive performance indicator. In all n_{cand} settings, *cTaskMat* has the best performance on F1-score.

In addition, we evaluate the system performance by varying the task–worker ratio from 2 to 10. We fix the number of workers as 168 and increase the number of total tasks. When the task–worker ratio increases, the total number of tasks that can be assigned by the system also increases proportionally. This impact alone tends to decrease the "coverage" proportionally. For instance, compared with the ratio of 2, the coverage should drop to 1/5 when the ratio is increased to 10. However, because coverage is defined as the ratio between all the selected tasks by workers over all the tasks that can be assigned in the

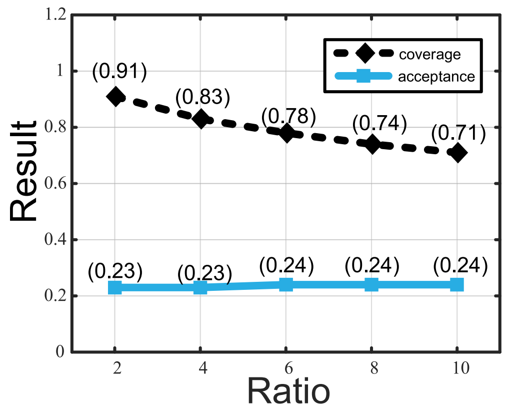

Fig. 9. System performance by varying the task–worker ratio.

system, from Fig. 9, we can see that the decreasing of coverage is much slower than that speed. This is because when the task–worker ratio is increased, more tasks are available as well. In this case, workers have more chances to select distinct tasks, which will also increase the number of all the selected tasks by workers. Therefore, from this experiment, we find that the task–worker ratio could decrease coverage in a moderate level, e.g., when the ratio is up to 10, the coverage can be still more than 71% in Fig. 9. On the other hand, since each worker selects tasks from the candidate task list only, the acceptance rate is thus relatively stable across different task–worker ratios.

In summary, these experimental results demonstrate that *cTaskMat* well balances different metrics and has a more comprehensive performance for real applications.

V. Related Work

The wide popularity of mobile devices (e.g., smartphones and wearables) has brought a novel sensing and computing paradigm named *mobile crowdsensing* [7], which outsources a large-scale complex job to the crowd having mobile devices to perform the location-dependent task at large scale. In recent years, we have witnessed the successes of mobile crowdsensing in a plenty of smart city applications, such as traffic monitoring [16], [46], air quality monitoring [44], urban noise mapping [22], intelligent transit services [3], [18], [41], etc. In order to guarantee the effectiveness and efficiency of mobile crowdsensing applications, other factors of crowdsensing, including incentive mechanism [33], [45], security and privacy protection [12], [13], [19], [26], and accuracy of data collected [43], have also been widely explored. Specifically, Meng *et al.* [19] analyzed the security and privacy problems in urban sensing. Li *et al.* [13] studied the privacy leakage of location sharing in mobile computing. Zhou *et al.* [42] proposed a method to recover data from collected data. These works demonstrate and ensure the ability of mobile crowdsensing on addressing large-scale practical problems.

Task assignment, or worker–task matching, is an important problem in the crowdsensing systems and has attracted a lot of research efforts. Difallah *et al.* [6] proposed a content-based task assignment approach that focuses on dynamically pushing tasks to workers so that to maximize the overall task acceptance rate. He *et al.* [10] considered the optimal task allocation problem in location-dependent crowdsensing. Liu *et al.* [17] proposed a unified task assignment design for urban sensing, which comprehensively considers the coverage, latency, and accuracy of task assignment to optimize the overall system utility. Karaliopoulos *et al.* [11] estimated each worker's probability of accepting a task through the logistic regression technique and made use of such information to match tasks with workers. Yang *et al.* [35] studied the personalized task recommendation by considering workers' sensing reliability in crowdsensing. None of these works, however, comprehensively consider workers' task preferences and their attitudes on task attributes during the worker–task matching. In contrary, *cTaskMat* extensively mines such information from historical worker behaviors and exploits their task preferences for better task assignments to balance performance metrics and reduce the system cost.

The incentive mechanism design is another important issue in crowdsensing and will decide the system costs. Zhao *et al.* [40] considered the problem of the budget-constrained incentive mechanism design for crowdsensing in both offline and online scenarios. Xu *et al.* [32] proposed a cost-saving method to lower user burdens. Singer and Mittal [27] and Singla and Krause [28] presented pricing mechanisms for crowdsourcing markets that rely on the bidding model and the posted pricing model. These works need explicit feedback from workers or make some unrealistic assumptions on workers' preferences. Lin *et al.* [14] proposed an implicit feedback-based recommendation system for crowdsensing. However, their work mainly focuses on the optimization of task acceptance and omits the concerns on task coverage and system cost. *cTaskMat* aims to address the misaligned task coverage problem in crowdsensing, and meanwhile, optimize other performance metrics.

VI. Conclusion

This article presented *cTaskMat* for coverage-oriented task assignments in mobile crowdsensing systems. *cTaskMat* implicitly learns workers' task preferences and their attitudes on task attributes, and exploits such information to achieve better task assignments. The experimental results using the real-world experiment and large data set-driven evaluation demonstrate that *cTaskMat* can achieve comprehensive performances on task acceptance and task coverage, while reducing the cost.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[2] A. Ahmed and E. Xing, "Scalable dynamic nonparametric bayesian models of content and users," in *Proc. 23rd Int. Joint Conf. Artif. Intell. (IJCAI)*, 2013, pp. 3111–3115.

[3] C. Cao, Z. Liu, M. Li, W. Wang, and Z. Qin, "Walkway discovery from large scale crowdsensing," in *Proc. ACM/IEEE Int. Conf. Inf. Process. Sens. Netw. (IPSN)*, Porto, Portugal, 2018, pp. 13–24.

[4] J. Chen, H. Zhang, X. He, L. Nie, and T. S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 335–344.

[5] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T. S. Chua, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 5659–5667.

[6] D. E. Difallah, G. Demartini, and P. Cudré-Mauroux, "Pick-a-Crowd: Tell me what you like, and i'll tell you what to do," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, 2013, pp. 367–374.

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

[8] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.

[9] J. He, X. Li, and L. Liao, "Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 1837–1843.

[10] S. He, D. H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Toronto, ON, Canada, 2014, pp. 745–753.

[11] M. Karaliopoulos, I. Koutsopoulos, and M. Titsias, "First learn then earn: Optimizing mobile crowdsensing campaigns through data-driven user profiling," in *Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2016, pp. 271–280.

[12] H. Li, H. Zhu, S. Du, X. Liang, and X. S. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 4, pp. 646–660, Jul./Aug. 2018.

[13] H. Li, H. Zhu, and D. Ma, "Demographic information inference through meta-data analysis of Wi-Fi traffic," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1033–1047, May 2018.

[14] C. H. Lin, E. Kamar, and E. Horvitz, "Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 908–914.

[15] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[16] Z. Liu, S. Jiang, P. Zhou, and M. Li, "A participatory urban traffic monitoring system: The power of bus riders," *IEEE Trans. Intell. Transport. Syst.*, vol. 18, no. 10, pp. 2851–2864, Oct. 2017.

[17] Z. Liu, Z. Li, and K. Wu, "UniTask: A unified task assignment design for mobile crowdsourcing-based urban sensing," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6629–6641, Aug. 2019.

[18] Z. Liu, Z. Li, K. Wu, and M. Li, "Urban traffic prediction from mobility data using deep learning," *IEEE Netw. Mag.*, vol. 32, no. 4, pp. 40–46, Jul./Aug. 2018.

[19] Y. Meng, W. Zhang, H. Zhu, and X. S. Shen, "Securing consumer IoT in the smart home: Architecture, challenges, and countermeasures," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 53–59, Dec. 2018.

[20] S. Morishita *et al.*, "SakuraSensor: Quasi-realtime cherry-lined roads detection through participatory video sensing by cars," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, 2015, pp. 695–705.

[21] R. W. Ouyang, M. Srivastava, A. Toniolo, and T. J. Norman, "Truth discovery in crowdsourced detection of spatial events," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1047–1060, Apr. 2016.

[22] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: An end-to-end participatory urban noise mapping system," in *Proc. ACM/IEEE Int. Conf. Inf. Process. Sens. Netw. (IPSN)*, 2010, pp. 105–116.

[23] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell. (UAI)*, 2009, pp. 452–461.

[24] F. Ricci, B. Shapira, and L. Rokach, *Recommender Systems Handbook*, 2nd ed. Boston, MA, USA: Springer, 2015, pp. 1–1003, doi: 10.1007/978-1-4899-7637-6.

[25] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, 2001, pp. 285–295.

[26] J. Shu, X. Liu, X. Jia, K. Yang, and R. H. Deng, "Anonymous privacy-preserving task matching in crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3068–3078, Aug. 2018.

[27] Y. Singer and M. Mittal, "Pricing mechanisms for crowdsourcing markets," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, 2013, pp. 1157–1166.

[28] A. Singla and A. Krause, "Truthful incentives in crowdsourcing tasks using regret minimization mechanisms," in *Proc. 22nd Int. Conf. WWW*, 2013, pp. 1167–1178.

[29] A. Vaswani *et al.*, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS 2017)*, Long Beach, CA, USA, 2017, pp. 6000–6010.

[30] J. Wang *et al.*, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 515–524.

[31] Y. Wu, Y. Wang, and G. Cao, "Photo crowdsourcing for area coverage in resource constrained environments," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Atlanta, GA, USA, 2017, pp. 1–9.

[32] L. Xu, X. Hao, N. D. Lane, X. Liu, and T. Moscibroda, "More with less: Lowering user burden in mobile crowdsourcing through compressive sensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, 2015, pp. 659–670.

[33] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2012, pp. 173–184.

[34] D. Yang, G. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1732–1744, Jun. 2016.

[35] S. Yang, K. Han, Z. Zheng, S. Tang, and F. Wu, "Towards personalized task matching in mobile crowdsensing via fine-grained user profiling," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 2411–2419.

[36] M. Youssef, M. Youssef, A. Uchiyama, H. Yamaguchi, and T. Higashino, "TransitLabel: A crowd-sensing system for automatic labeling of transit stations semantics," in *Proc. 14th Annu. Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, 2016, pp. 193–206.

[37] D. Zhang, J. Huang, Y. Li, F. Zhang, C. Xu, and T. He, "Exploring human mobility with multi-source data at extremely large metropolitan scales," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2014, pp. 201–212.

[38] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1655–1661.

[39] M. Zhang and Y. Wu, "An unsupervised model with attention autoencoders for question retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4978–4986.

[40] D. Zhao, X. Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Toronto, ON, Canada, 2014, pp. 1213–1221.

[41] P. Zhou, Y. Zheng, and M. Li, "How long to wait? Predicting bus arrival time with mobile phone based participatory sensing," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1228–1241, Jun. 2014.

[42] T. Zhou, Z. Cai, K. Wu, Y. Chen, and M. Xu, "FIDC: A framework for improving data credibility in mobile crowdsensing," *Comput. Netw.*, vol. 120, pp. 157–169, Jun. 2017.

[43] T. Zhou, B. Xiao, Z. Cai, M. Xu, and X. Liu, "From uncertain photos to certain coverage: A novel photo selection approach to mobile crowdsensing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 1979–1987.

[44] Q. Zhu, M. Y. S. Uddin, N. Venkatasubramanian, and C.-H. Hsu, "Spatiotemporal scheduling for crowd augmented urban sensing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 1997–2005.

[45] X. Zhu, J. An, M. Yang, L. Xiang, Q. Yang, and X. Gui, "A fair incentive mechanism for crowdsourcing in crowd sensing," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1364–1372, Dec. 2016.

[46] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2289–2302, Nov. 2013.

Shiwei Song received the B.E. degree in optoelectronics information science and engineering from Xi'an University of Posts and Telecommunications, Xi'an, China, in 2017. He is currently pursuing the master's degree in computer science and technology with Northwest University, Xi'an.

His research interests include crowdsensing and recommendation system.

Zhidan Liu (Member, IEEE) received the B.E. degree in computer science and technology from Northeastern University, Xi'an, China, in 2009, and the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2014.

After that, he worked as a Research Fellow with Nanyang Technological University, Singapore. He is currently an Assistant Professor with Shenzhen University, Shenzhen, China. His research interests include distributed sensing and mobile computing, crowdsourcing, big data analytics, and urban computing.

Zhenjiang Li (Member, IEEE) received the B.E. degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2007, and the M.Phil. degree in electronic and computer engineering and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2009 and 2012, respectively.

He is currently an Assistant Professor with the Computer Science Department, City University of Hong Kong, Hong Kong. His research interests include distributed sensing, wearable and mobile computing, Internet of Things, cyber–physical security, and wireless and urban computing.

Tianzhang Xing (Member, IEEE) received the B.E. degree from the School of Telecommunications Engineering, Northwest University, Xi'an, China, and the Ph.D. degree from the School of Information and Technology, Northwest University.

He is currently an Associate Professor with the School of Information and Technology, Northwest University. His current research interests include mobile computing, pervasive computing, and wireless networks.

Dingyi Fang (Member, IEEE) received the Ph.D. degree in computer application technology from Northwestern Polytechnical University, Xi'an, China, in 2001.

He is currently a Professor with the School of Information Science and Technology, Northwest University, Xi'an. His current research interests include mobile computing and distributed computing systems, network and information security, and wireless sensor networks.