Multi-Task Allocation in Mobile Crowd Sensing with Individual Task Quality Assurance

Jiangtao Wang, Yasha Wang, Daqing Zhang, Feng Wang, Haoyi Xiong, *Member, IEEE*, Chao Chen, Qin Lv, and Zhaopeng Qiu

Abstract—Task allocation is a fundamental research issue in mobile crowd sensing. While earlier research focused mainly on single tasks, recent studies have started to investigate multi-task allocation, which considers the interdependency among multiple tasks. A common drawback shared by existing multi-task allocation approaches is that, although the overall utility of multiple tasks is optimized, the sensing quality of individual tasks may become poor as the number of tasks increases. To overcome this drawback, we re-define the multi-task allocation problem by introducing task-specific minimal sensing quality thresholds, with the objective of assigning an appropriate set of tasks to each worker such that the overall system utility is maximized. Our new problem also takes into account the maximum number of tasks allowed for each worker and the sensor availability of each mobile device. To solve this newly-defined problem, this paper proposes a novel multi-task allocation framework named MTasker. Different from previous approaches which start with an empty set and iteratively select task-worker pairs, MTasker adopts a descent greedy approach, where a quasi-optimal allocation plan is evolved by removing a set of task-worker pairs from the full set. Extensive evaluations based on real-world mobility traces show that MTasker outperforms the baseline methods under various settings, and our theoretical analysis proves that MTasker has a good approximation bound.

Index Terms—Mobile crowd sensing, multi-task allocation, sensing quality, submodular optimization

1 Introduction

Mobile crowd sensing (MCS) [1], [2] has become a new sensing paradigm, where mobile users (called workers) utilize their smart devices to sense and collect real-time information in their environments (e.g., air quality [3], traffic information [4], and noise level [5] in urban areas). Some MCS platforms have been developed (e.g., Campaignr [6], Medusa [7], AndWellness [25] and PRISM [24]), which allow workers to decide which tasks to complete by themselves. However, workers usually have to search many times and use manual filtering to find suitable tasks, which is time-consuming [8]. Therefore, task allocation [8], [9], [11] is a fundamental research issue for MCS platforms, in which MCS platforms automatically assign tasks to the appropriate workers.

- *J. Wang, D. Zhang, F. Wang, and Z. Qiu are with the School of EECS, Peking University, Beijing 100080, China, and the Key Laboratory of High Confidence Software Technologies, Ministry of Education. E-mail: jiangtao19871104@sina.com, daqing.zhang@telecom-sudparis.eu, {wangfeng2013, qiuzhaopeng}@pku.edu.cn.*
- *Y. Wang is with the Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China, and the National Engineering Research Center of Software Engineering, Peking University, Beijing 100871, China. E-mail: yasha.wang@163.com.*
- *H. Xiong is with the Missouri University of Science and Technology, Rolla, MO 65409. E-mail: xhyccc@gmail.com.*
- *C. Chen is with the College of Computer Science, Chongqing University, Chongqing 400044, China. E-mail: cschaochen@cqu.edu.cn.*
- *Q. Lv is with the University of Colorado Boulder, Boulder, CO 80309. E-mail: qin.lv@colorado.edu.*

Manuscript received 30 June 2017; revised 26 Nov. 2017; accepted 9 Jan. 2018. Date of publication 15 Jan. 2018; date of current version 2 Aug. 2018. (Corresponding author: Yasha Wang.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2018.2793908

Recently, there have been many studies on MCS task allocation [11], [12], [14], [29], [30], [31]. In the earlier stage of MCS research, existing approaches (e.g., [11], [12], [14], [16], [23], [39]) are mostly single-task oriented, where they assume that tasks on MCS platforms are isolated, so that the task allocation is executed for each single task independently.

However, as the number of MCS tasks increases, the tasks are no longer independent, because they compete with each other in a shared and limited resource pool (e.g., shared user pool or total budget). Thus, in order to better coordinate tasks and make full use of the limited resources, some recent studies (e.g., [29], [30], [31]) have started to focus on multi-task allocation, where the interdependency of multiple tasks is considered. Typically, the objective is to optimize the overall utility of multiple tasks. For example, Wang et al. [33] studied the overall utility maximization of multiple tasks with workers sensing capability constraints, while Song et al. [31] and Wang et al. [32] proposed frameworks to optimize the overall utility with a total incentive budget constraint. In these works, the overall utility is all defined as the weighted sum of each tasks sensing quality (e.g., temporal-spatial coverage).

Although these methods (e.g., [31], [32], [33]) have been shown to be effective in optimizing the overall utility of multiple tasks with certain constraints, they do not take the sensing quality of each individual task into consideration. Essentially, in these works, the optimization of overall utility is to achieve a trade-off among the sensing quality of multiple tasks when the shared resource (e.g., number of workers or budget) is limited. Intuitively, as the number of MCS tasks increases, the resource competition among

TABLE 1
Task Attributes in the Example
(The Weights of Three Tasks are Equal)

	t_1	t_2	t_3
temporal-spatial coverage	0.6	0.7	0.75
minimum required coverage	0.55	0.75	0.7
simple weighted average		(0.6+0.7+0.75)/3	
utility defined by MTasker		(0.6+0+0.75)/3	

multiple tasks becomes more intense. Thus, under such circumstances, *even if the overall utility is optimized by a certain task allocation strategy, the sensing quality of individual tasks may turn out to be poor.* Since the ultimate goal of data collection in smart cities through MCS is to provide relevant services or applications, the poor quality of sensing tasks will diminish the usability of these services or applications, or even make them useless. For example, imagine a smart city navigation service that is developed based on the traffic congestion information collected through an MCS task. If the sensing quality is poor (e.g., only with 20 percent temporal-spatial coverage), the navigation service may fail to find optimal travel routes for its users, because the traffic information in many relevant regions are unknown. To make matters worse, when the services are useless due to the low quality of sensing data, efforts made by MCS workers and consumed computation resources (e.g., battery consumption [40]) have already been wasted without the opportunity to revert the action. As such, *optimizing the overall utility without considering the sensing quality of individual tasks can have an adverse impact on the performance of multi-task MCS platforms.*

To overcome this limitation, this paper studies a novel multi-task allocation problem with quality-assurance on each individual task, in which task organizers pre-define *task-specific minimum sensing quality thresholds for each task.* Specifically, the sensing quality of each task is measured by the temporal-spatial coverage, which is widely used in the state-of-the-art research works [10], [27], [28]. Different from existing research works for multi-task allocation (e.g., [31], [32], [33]), if the obtained temporal-spatial coverage of a certain task is lower than its threshold, it is considered to contribute nothing to the overall utility. For example, consider three equally-important tasks t_1, t_2 and t_3 with obtained temporal-spatial coverage of 0.6, 0.7 and 0.75. If we use the utility functions defined in [31], [32], [33], then the achieved overall utility is computed as (0.6+0.7+0.75)/3. However, when considering the quality assurance on individual tasks, if the minimum required coverage of t_1, t_2 and t_3 are given as 0.55, 0.75 and 0.7, respectively, then the overall utility would be calculated as (0.6+0+0.75)/3 and the utility of t_2 is disregarded (see Table 1). In this paper, we aim to assign an appropriate set of tasks to each worker in order to maximize the overall utility while meeting the minimal quality requirements of individual tasks. In addition, we consider the following two practical constraints. First, a certain worker is only able to complete a subset of tasks, because his/her mobile device may not be embedded with required sensors. Second, to avoid task overloading, each worker pre-defines the maximum number of sensing tasks he/she can accept.

Since the multi-task allocation problem for MCS is NP-hard in nature [30], we cannot solve it by enumerating all possible combinations. Thus, existing approaches for multi-task allocation [30], [31], [32], [33], though with different constraints or utility function definitions, commonly use greedy-based algorithms to iteratively select task-worker pairs. Specifically, these algorithms incrementally select the next task-worker pair with the maximal utility until a stopping criterion has been triggered (e.g., budget has been used up, or all valid task-worker pairs have been assigned). However, because of the newly introduced minimal sensing quality thresholds of individual tasks, state-of-the-art approaches cannot be directly adopted in our multi-task allocation problem. The reason is that during earlier iterations of the greedy selection process, it is likely that none of the task-worker pairs can lead to an overall utility increase when adding to the set, as the tasks minimum threshold has not been reached.

To tackle the above mismatch between existing task allocation algorithms and our re-defined problem, one may think of the following two methods. One direct adjustment is to randomly select task-worker pairs in earlier iterations before the sensing quality of one task reaches the minimum threshold. With the additions of more task-worker pairs, tasks with lower thresholds would reach their thresholds earlier, so that the algorithm tends to allocate more limited resources (i.e., workers) to those tasks. However, this characteristic in the relationship between the thresholds and allocated resources may have a negative impact on the overall utility optimization. For example, when tasks with lower thresholds are less important (i.e., with lower weights) and total resources are limited, important tasks (i.e., tasks with higher weights) are less likely to be assigned workers, which is bad for the overall utility maximization. Another naïve method is to use a relaxed utility function, which regards the minimum thresholds as zero when selecting task-worker pairs. However, when the total worker resource is limited, the obtained sensing quality of some tasks may be lower than their thresholds, so that the allocated workers for these tasks are wasted. In summary, *direct modification or adjustment of existing algorithms is not appropriate to solve our targeted multi-task allocation problem, and a new approach is needed.*

With the aforementioned research objective and challenges, this paper presents MTasker, a framework for MCS multi-task allocation. Our main contributions are summarized as follows:

(1) *We investigate the drawback of existing solutions for multi-task allocation in MCS, and study a novel problem by introducing task-specific minimum sensing quality thresholds for each task.* Keeping the thresholds in mind, this paper aims to assign an appropriate set of tasks to each worker in order to maximize the overall utility of multiple tasks, while considering two worker-side constraints (i.e., maximum number of sensing tasks allowed for each worker and the sensor availability of each mobile device). To the best of our knowledge, we are the first to consider quality thresholds on individual tasks in multi-task allocation for MCS.

(2) *We propose an offline multi-task allocation framework, called MTasker.* Different from existing approaches which start from an empty set and iteratively select/add task-worker pairs into the set, MTasker adopts a

descent greedy approach which searches quasi-optimal allocation by removing a set of task-worker pairs from the full set. MTasker consists of the following two key steps. It first pseudo-allocates task-worker pairs and checks the validity of some pairs, based on which the original overall utility maximization problem is transformed into removing a set of task-worker pairs to minimize the overall utility reduction. It then takes an iterative greedy process to optimize this transformed problem based on the theory of submodular function optimization. Theoretical analysis shows that MTasker can achieve quasi-optimality globally with low computation complexity.

(3) *We evaluate MTasker extensively with real-world datasets that contain mobility traces of more than 50,000 mobile users.* The results verify that, under various settings, MTasker outperforms baseline methods by achieving higher overall system utility. We further conduct detailed analysis to understand why MTasker outperforms the other methods.

2 Related Work

Many online crowdsourcing platforms (e.g., [17], [18], [19]) have been developed in recent years. For example, Mturk [17] is a well-known online crowdsourcing platform, which has attracted many organizers to publish tasks and plenty of workers to complete tasks. Meanwhile, automatic task allocation problem is also extensively studied for online crowdsourcing, and corresponding approaches [20], [21], [22] are proposed with different optimization goals and constraints. However, these task allocation approaches are designed for human intelligence tasks (e.g., image classification tasks in Mturk), which can be completed online even through the desktop computer. On the contrary, the goal of MCS is to sense physical environment or phenomenon (e.g., temperature, traffic status and air quality) in smart cities, which is location-dependent and requires workers' physical presence at specific locations.

Automatic task allocation is a key research issue in MCS and has attracted much attention in recent years. In [26], [27] Reddy et al. studied worker recruitment in participatory sensing, and proposed a coverage-based recruitment strategy to select a predefined number of workers so as to maximize the spatial coverage. Cardone et al. [28] developed a MCS platform, where a simple worker selection mechanism was proposed to maximize the spatial coverage of crowdsensing with predefined number of workers. Singla et al. [13] proposed a novel adaptive worker selection mechanism for maximizing spatial coverage under total incentive constraint in community sensing with respect to privacy. In [11], [12] the authors considered another version of the task assignment problem, where the optimization goal is to maximize the coverage quality under an overall budget constraint. All these research works focused on singe talk allocation and did not consider the interdependency among multiple tasks.

In recent years, some efforts have been made in the area of multi-task allocation. Guo et al. [15] study the multi-task allocation problem under two scenarios, i.e., the time-sensitive and time-insensitive. Xiao et al. [29] studied the task assignment problem in mobile social networks, and their objective is to minimize the average makespan. Liu et al. [30] studied

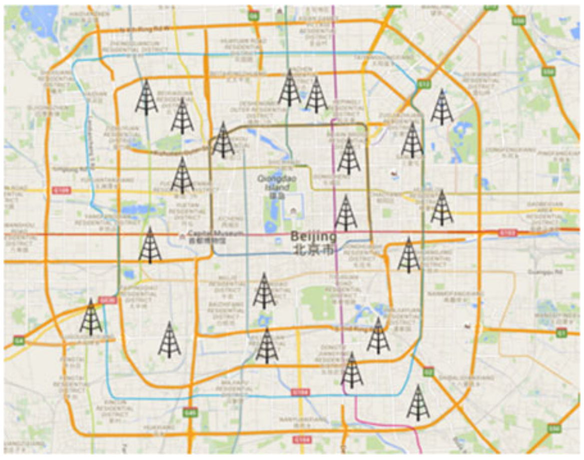


Fig. 1. Target sensing area: An example.

the task allocation problem when considering the relationship between the number of workers and the number of tasks to be completed. The optimization goals and constraints of [29], [30] are significantly different from our work. The research works that are closest to ours are [31], [32], [33], where the authors attempt to optimize the overall utility when multiple tasks share a limited resource pool. Both [31], [32] proposed multi-task allocation algorithms to maximize overall system utility when tasks share a total budget. The multi-task allocation strategy proposed in [33] aims at optimizing the overall utility when multiple tasks share a limited pool of workers. Although these studies did consider the interdependency of multiple tasks, their objective is only to optimize overall utility without considering the sensing quality of each task. In contrast, our work differs from them in the following two aspects. (1) In terms of problem definition, we re-define the multi-task allocation problem by introducing task-specific minimum sensing quality thresholds. (2) In terms of task allocation algorithm, instead of adding workers or task-worker pairs one by one from an empty set, MTasker adopts a descent greedy approach, where a quasi-optimal allocation plan is evolved through the reduction from a full set of task-worker pairs.

3 Problem Analysis and Formulation

In this section, we first give a motivating example, and then define the system models and present relevant assumptions. Finally, we formulate the multi-task allocation problem.

(Example 1). CrowdSense is an MCS-based urban information sharing platform. Let us consider a downtown area with sensing requirements for a certain time period (e.g., one week). This downtown area is divided into several virtual subareas, and the region covered by each cell tower is regarded as a subarea (see Fig. 1). The whole sensing time period is divided into equal-length cycles (e.g., one hour per cycle). There are a number of (e.g., 10 tasks) concurrent sensing tasks running on the platform (e.g., air quality monitoring, traffic congestion detection, etc.) sharing the same subareas and cycles. The sensing quality of each task is measured by the temporal-spatial coverage which is widely used in state-of-the-art research works [10], [27], [28]. Each task is published by an organizer with a task-specific minimum temporal-spatial coverage threshold. Each worker pre-defines his/her maximum number of assigned tasks through the CrowdSense mobile client. Besides, the mobile phone of each worker has a set of sensors, which can complete different sets of MCS tasks.

In this example use case, our objective is to select a subset of task-worker pairs for maximizing the overall utility for CrowdSense, while meeting the minimum temporal-spatial coverage threshold of each task. We denote the set of MCS tasks published by task organizers as $T = \{t_1, t_2 \ldots t_r\}$, and the set of candidate workers as $W = \{w_1, w_2 \ldots w_l\}$. We denote all task-worker pairs as $V = W \times T = \{(w_u, t_k)|w_u \in W,\ t_k \in T\}$. Based on the availability of sensors, we further select valid task-worker pairs from V and denote them as $V_f = \{(w_u, t_k)|w_u \in W,\ t_k \in T,\ and\ the\ mobile\ device\ of\ w_u\ can\ complete\ t_k\}$.

When publishing the tasks onto the MCS platform, task organizers pre-define the following task-specific attributes. As illustrated in this example use case, this paper focuses on distributed environment monitoring tasks. We use the temporal-spatial coverage as the metric to characterize the sensing quality, which is widely used in state-of-the-art works (e.g., [10], [27], [28]). Specifically, the organizer specifies the target sensing area consisting of a set of subareas, denoted as $S = \{s_1, s_2 \ldots s_i \ldots s_m\}$, and sensing duration which is divided into equal-length cycles, denoted as $C = \{c_1, c_2 \ldots c_j \ldots c_n\}$. Then, the temporal-spatial coverage of task t_k is measured as the fraction of temporal-spatial cells covered by V_f denoted as $\frac{|Covered(t_k,V_f)|}{|S\times C|}$, where $Covered(t_k, V_f)$ is the set of temporal-spatial cells with at least one sensor reading for task t_k. The task organizer of task t_k pre-defines the minimum temporal-spatial coverage threshold as $MT(t_k)$ according to his/her the domain expertise.

Specifically, the overall utility achieved by a set of valid task-worker pairs V_f s defined as follows:

$$U(V_f) = \sum_{k=1}^{k=r} u_k(V_f) * wt_k, \tag{1}$$

where the weight factor $wt_k \in (0, 1)$ characterizes the importance of task t_k and $u_k(V_f)$ is further defined as follows:

$$u_k(V_f) = \frac{|Covered(t_k, V_f)|}{|S \times C|}, if \frac{|Covered(t_k, V_f)|}{|S \times C|} \geq MT(t_k) \tag{2}$$

$$u_k(V_f) = 0, if \frac{|Covered(t_k, V_f)|}{|S \times C|} < MT(t_k). \tag{3}$$

From Eqs. (2), (3) we can see that, if the obtained temporal-spatial coverage is lower than the minimum threshold, it contributes nothing to the overall utility of the multi-task MCS platform.

The objective of MTasker is to select an appropriate set of task-worker pairs to maximize the overall utility while satisfying the maximum number of tasks allowed for each worker and the sensor availability of each mobile device. Therefore, it can be formulated as selecting a subset from V_f denoted as $V_f^{'} \subseteq V_f (if (w_u, t_k) \in V_f^{'})$, then we assign task t_k to worker w_u to maximize the overall utility defined in Eq. (1), subject to the maximum number of allowed tasks. The formulation is presented in Eqs. (4), (5).

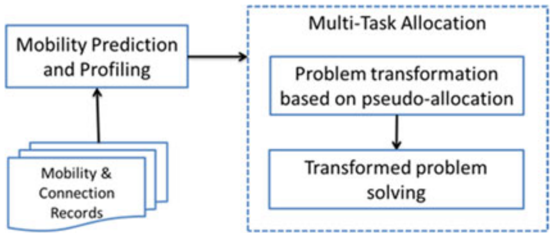

Mobility Prediction and Profiling
Mobility & Connection Records
Multi-Task Allocation
Problem transformation based on pseudo-allocation
Transformed problem solving

Fig. 2. MTasker: framework overview.

$$Maximize \sum_{k=1}^{k=r} V_f^{'} * wt_k \tag{4}$$

$$Subject\ to \sum_{k=1}^{k=r} y_{u,k} \leq L_u, \tag{5}$$

where $y_{u,k} = 1$ if task t_k has been assigned to w_u, and $y_{u,k} = 0$ otherwise. L_u is the maximum number of assigned tasks pre-defined by worker w_u.

4 MTasker Framework

In this section, we first present an overview of the MTasker framework design. Then the major components are described in detail.

4.1 Framework Overview

Our framework, named MTasker (see Fig. 2), follows a centralized design, where a central server collects and stores the volunteering mobile users historical call traces in the target sensing area, and the server selects task-worker pairs. The assignments are pre-downloaded before an MCS task is executed. The server pushes the assigned tasks to the workers when he/she connects to the cellular tower [10]. Two major components of MTasker are described as follows.

- *Mobility Prediction and Profiling.* This component predicts each workers mobility and connection to the cellular tower, using the historical data collected from the telecom operator. Specifically, it estimates the probability $Pro_{u,i,j}$ of worker w_u connecting at least once to each subarea (cell tower) s_i during the sensing cycle c_j.
- *Multi-task Allocation.* This component selects a subset of task-worker pairs based on the predicted mobility, the optimization goal, and constraints. Different from existing approaches [31], [32], [33] which add task-worker pairs one by one starting from an empty set, MTasker adopts a descent greedy approach which searches for a quasi-optimal allocation by removing task-worker pairs from the full set, and it consists of the following two phases. *(1) Problem transformation based on pseudo-allocation.* In this phase, MTasker pseudo-allocates all task-worker pairs, and then constructs a full set of task-worker pairs after validity checking and task deletion. The original overall utility maximization problem is thus transformed into removing a set of task-worker pairs from the full set to minimize the overall utility reduction. *(2) Transformed problem solving.* In this phase, MTasker uses an iterative greedy process to optimize this transformed

problem based on the theory of submodular set function optimization. Specifically, it removes a set of task-worker pairs in order to minimize the overall utility reduction, while satisfying the constraints on each workers maximum number of assigned tasks.

4.2 Mobility Prediction and Profiling

This component predicts the number of samples during cycle c_j at subarea s_i by calculating the probability of each worker connecting to the tower at least one time in each sensing cycle. We first map each workers historical call traces onto sensing cycles. Then we count the average number of connections by each worker w_u at each cell tower s_i in each cycle c_j, which is denoted as $\lambda_{u,i,j}$.

For example, to estimate $\lambda_{u,i,j}$ for sensing cycle c_j from 08:00 to 09:00 of a specific day, we count the average number of connections by w_u at s_i during the same period in the historical mobility and connection records. Assuming that the connection sequence follows an inhomogeneous Poisson process [10], [11], the probability of worker w_u to connect h times to cell tower s_i in sensing cycle c_j can be modeled as

$$\varphi_{u,i,j}(h) = \lambda_{u,i,j}^{h} * e^{-\lambda_{u,i,j}}/h!. \quad (6)$$

Therefore, we can estimate the probability of worker w_u connecting at least once during cycle c_j at s_i as follows:

$$Pro_{u,i,j} = \sum_{h=1}^{\infty} \varphi_{u,i,j}(h) = 1 - e^{-\lambda_{u,i,j}}, \quad (7)$$

Thus we predict the probability of worker w_u providing one sample for each assigned task during cycle c_j at s_i as:

$$\alpha_{i,j}(w_u) = 1 - e^{-\lambda_{u,i,j}}. \quad (8)$$

4.3 Multi-Task Allocation

With the minimum temporal-spatial coverage thresholds in mind, this component adopts a descent greedy approach, where a quasi-optimal allocation plan is evolved by removing a set of task-worker pairs from the full set. The core algorithm mainly consists of the following two phases.

4.3.1 *Problem Transformation Based on Pseudo-Allocation*

Given a full set of feasible task-worker pairs $V_f = \{(w_u, t_k) | w_u \in W,\ t_k \in T,\ \textit{and the mobile device of } w_u \textit{ can complete } t_k\}$, MTasker first estimates the temporal-spatial coverage if the constraints on the maximum number of assigned task is removed. Specifically, we need to estimate the achieved temporal-spatial coverage of each task t_k (denoted as $\frac{|Covered(t_k,V_f)|}{|S\times C|}$) as follows:

(1) Given a specific temporal-spatial cell (s_i, c_j) for task t_k, we calculate its probability to be covered as follows:

$$\beta_{i,j}^{k} = 1 - \prod_{(w_u,t_k)\in V_f, w_u\in W} (1 - \alpha_{i,j}(w_u)). \quad (9)$$

(2) Then the temporal-spatial coverage of task t_k achieved by the full set of feasible task-worker pairs V_f is estimated as

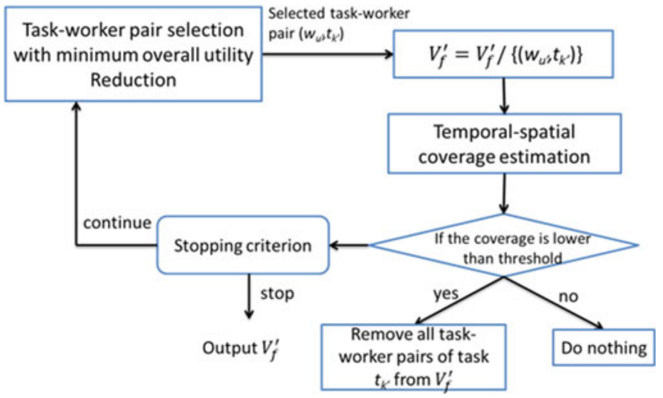

Task-worker pair selection with minimum overall utility Reduction
Selected task-worker pair $(w_{u'}, t_{k'})$
$V_f' = V_f' / \{(w_{u'}, t_{k'})\}$
Temporal-spatial coverage estimation
If the coverage is lower than threshold
yes
no
Stopping criterion
continue
stop
Output V_f'
Remove all task-worker pairs of task $t_{k'}$ from V_f'
Do nothing

Fig. 3. Algorithm workflow for solving the transformed problem.

$$\frac{|Covered(t_k, V_f)|}{|S \times C|} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} \beta_{i,j}^{k}}{m * n}. \quad (10)$$

If the temporal-spatial coverage of task t_k If the temporal-spatial coverage of task (w_u, t_k) from V_f, and the remaining task-worker pairs form a set denoted as $V_{f(1)}$. The intuition behind this operation is that if the minimum temporal-spatial coverage threshold of a task t_k cannot be satisfied even with all possible resources allocated, then MTasker should not allocate any resources to it.

Then, we check $V_{f(1)}$ and calculate the number of assigned tasks for each worker w_u, which is denoted as $\xi(w_u)$. If $\xi(w_u) \le L_u$, then we select all its task-worker pairs of w_u and assign corresponding tasks to the worker (i.e., adding selected pairs into $V_0^{'}$). After the tasks are assigned, we remove these workers and assigned task-workers from $V_{f(1)}$. The intuition behind the above operation is that if a worker's maximum number of assigned tasks is not less than the number of tasks he/she is capable of completing according to the availability of sensors, then just let him/her complete all feasible tasks.

Therefore, the original problem (see Eqs. (4), (5)), i.e., selecting a subset of task-worker pairs to maximize the overall utility, is transformed into *minimizing the overall utility reduction by removing a set of task-worker pairs V_r from $V_{f(1)}$* while ensuring the maximum number of assigned tasks for each worker. The transformed problem is formulated as follows:

$$Minimize \left\{ \sum_{k=1}^{k=r} u_k(V_{f(1)}) * wt_k - \sum_{k=1}^{k=r} u_k(V_{f(1)} \backslash V_r) * wt_k \right\} \quad (11)$$

$$Subject\ to\ \sum_{k=1}^{k=r} y_{u,k} \le L_u, \quad (12)$$

where $V_{f(1)} \backslash V_r$ is the remaining set of task-workers pairs after removing V_r from $V_{f(1)}$.

4.3.2 *Transformed Problem Solving*

Given the predicted mobility of each worker, an iterative greedy algorithm is presented to select task-worker pairs incrementally to obtain the quasi-optimal set V_r. The workflow of the algorithm is presented in Fig. 3.

From Fig. 3 we can see that the algorithms workflow consists of the following steps:

(1) The algorithm first picks the single task-worker pair (w_u, t_k) in $V_{f(1)}$ causing minimum overall utility reduction when removing it from $V_{f(1)}$. Then, $V_f^{'} = V_{f(1)}$ (w_u, t_k), and it re-estimates the temporal-spatial coverage of task t_k based on Eqs. (9) and (10). If the estimated coverage is lower than the minimum threshold, then remove all task-worker pairs of t_k from $V_f^{'}$.

(2) The algorithm then selects another task-worker pair $(w_{u'}, t_{k'})$ causing minimum overall utility reduction when removing it from $V_f^{'}$. Then, $V_f^{'} = V_{f(1)}(w_{u'}, t_{k'})$, and we re-estimate the temporal-spatial coverage of task $t_{k'}$ based on Eqs. (9) and (10). If the estimated coverage is lower than the minimum threshold, then remove all task-worker pairs of task $t_{k'}$ from $V_f^{'}$.

(3) The algorithm keeps selecting and removing task-worker pairs until the number of assigned tasks for each worker satisfies the constraint (i.e., stopping criterion in Fig. 3). The final $V_f^{'}$ combined with $V_0^{'}$ (i.e., $V_f^{'} \bigcup V_0^{'}$) is the task allocation plan achieved by MTasker.

4.4 Algorithm Analysis

In this section, we analyze the proposed algorithm of MTasker. First, we propose a brute force approach that can find optimal multi-task allocation, and analyze its time complexity. Then, we show how MTasker could approximate the optimal solution with much lower and acceptable computation complexity.

Intuitively, it is easy to think of a brute force approach as follows. Given a set of task T and a set of workers W, the brute force algorithm enumerates all possible combinations of task-worker pairs that satisfy the maximum number of assigned tasks and sensor availability of each worker. For each combination, it estimates the overall utility based on Eqs. (1) and (9), (10). Finally, the combination with the maximum overall utility is the optimal solution. However, the brute force algorithm is time-consuming for large $|T|$ and $|W|$ values. For example, for 20 tasks and 5,000 workers, it must enumerate 2^{100000} possible combinations, which is unacceptable. Thus, we need a solution that approximates the optimal result with much lower computation complexity. The proposed MTasker approach adopts an iterative process. The running time complexity of the greedy algorithm is $O(|T| * |W| * (|T| - \min\{L_u\}))$. In the worst case, the running time is $O(|T|^2 * |W|)$. The time complexity of MTasker is similar to the greedy-based algorithms proposed in [11], [31], [33].

First of all, we have to keep in mind that the original objective is to select a subset $V_f^{'}$ from all feasible task-worker pairs $V_{f(1)}$ (i.e., $V_f^{'} \subseteq V_{f(1)}$) for maximizing $U(V_f^{'}) = \sum_{k=1}^{k=r} u_k(V_f^{'}) * wt_k$. Then, we have transformed it into selecting and removing a subset $V_r \subseteq V_{f(1)}$, and the goal is accordingly transformed into minimizing the overall utility reduction (see Eq. (11)). We can re-write the utility reduction function in (11) as $f(V_r) = U(V_{f(1)}) - U(V_{f(1)}V_r)$. From the definition of $f(V_r)$, we can see that the first part (i.e., $U(V_{f(1)})$ is a constant value, so that minimizing $f(V_r)$ can be transformed into maximizing $h(V_r) = U(V_{f(1)}V_r)$. We further prove that the objective function $h(V_r)$ is a submodular set function (see Proof 1). Suppose the optimal solution of the problem selects a set of edges V^* to prune, while our Greedy solution selecting edges denoted as V_G. We here discuss the performance in the following two cases: *(1) optimal solution removes more edges than greedy* (i.e.,$|V_G| \leq |V^*|$), *and (2) optimal solution removes less edges than greedy* (i.e.,$|V^*| \leq |V_G|$).

(1) When $|V_G| \leq |V^*|$ we first assume that, using our Greedy algorithm, we select a same number of edges as $|V^*|$, we denote such set of edges as $|V_G^*|$. According to the submodularity of the utility function $h(\cdot)$ and literature [34], there exists

$$h(V_G^*) \geq (1 - e^{-1}) * h(V^*).$$

As both V_G and V_G^* are both selected by the same Greedy search using same utility function and $|V_G^*| = |V^*| \geq |V_G|$, there exists $V_G \subseteq V_G^*$.

Further, the utility function $h(\cdot)$ is a non-increasing monotonous submodular function. Thus, according to the non-increasing property, there exists

$$h(V_G) \geq h(V_G^*) \geq (1 - e^{-1}) * h(V^*).$$

In this case, we can conclude that our algorithm is near-optimal with $1 - e^{-1}$ bound for utility function $h(\cdot)$.

(2) When $|V^*| \leq |V_G|$, before the pair pruning, the full set of task-participant pairs is $V_{f(1)}$. Considering the total capacity of all users $\sum_{w_u \in W} L_u$, it is reasonable to conclude that, to ensure each worker satisfies the maximum number of allowed tasks constraints, there will be at least to remove edges K_f, where

$$K_f = |V_{f(1)}| - \sum_{w_u \in W} L_u.$$

Thus, we can easily get $|V_G| \geq |V^*| \geq K_f$. We assume there exists an optimal set of pairs V_{kf}, which consists of K_f pairs and maximize the utility function, i.e., $V_{kf} = \mathrm{argmax}_{|V|=K_f} h(V)$. Then, for all K_f-length subset of V^* denoted as $V_{kf}^* \subseteq V^*$ and $|V_{kf}^*| = K_f$, according to the non-increasing property and $|V_G| \geq |V^*| \geq K_f$, there exists $h(V^*) \leq h(V_{kf}^*) \leq h(V_{kf})$. Further, we denote V_{Gkf} as a set of pairs selected by Greedy search and the length of V_{Gkf} is $|V_{Gkf}| = K_f$. Then we can conclude

$$h(V^*) \leq h(V_{kf}^*) \leq h(V_{kf}) \leq h(V_{Gkf})/(1 - e^{-1}).$$

Thus, we can conclude

$$h(V_G) \geq (1 - e^{-1})h(V^*) + \varepsilon,$$

where $\varepsilon = \{h(V_G) - h(V_{Gkf})\}$. Please note that, for any completed greedy search, the value ε is known. $h(V_{Gkf})$ is the utility function value, when the greedy search took K_f pairs to remove in the process, while $h(V_G)$ is the utility function value of the final selected pairs. For example, when the $h(V_G) = 0.8$ and $h(V_{Gkf}) = 0.82$, hen our bound for set function $h(\cdot)$ will be $h(V_G) \geq (1 - e^{-1})h(V^*) - 0.02$.

However, the above approximation ratio for $h(V_r)$ does not directly mean that the original objective function $U(V_f^*) = \sum_{k=1}^{k=r} u_k(V_f^{'}) * wt_k$ is maximized. This is because the following two operations are adopted by MTasker. 1) OP-1: removing the tasks whose estimated temporal-spatial coverage is lower than the minimum threshold. 2) OP-2: If $\xi(w_u) \leq L_u$, then we select all its task-worker pairs of w_u and assign corresponding tasks to the worker. In order to investigate if the original multi-task allocation problem is globally quasi-optimal, we further prove, in Proof 2 and Proof 3, that neither OP-1 nor OP-2 has a negative effect on the optimality. Besides, note that $U(V_{f(1)})$ in original objective function (i.e., $U(V_{f(1)}) - h(V_r)$) has constant value given a specific pool of workers and tasks, the above approximation bound is for the $h(V_r)$ indicates that the original objective function is quasi-optimal.

Proof 1. *The set function* $h(V_r) = \sum_{k=1}^{k=r} u_k(V_{f(1)} \backslash V_r) * wt_k$, $V_r \subseteq V_{f(1)}$ *is a submodular function*. In the greedy process, the algorithm iteratively removes task-worker pair (w_u, t_k) causing minimum overall utility reduction. If the re-estimated coverage of t_k is lower than the minimum threshold, MTasker deletes t_k and removes all task-worker pairs of t_k. From the above operation, we can see that during the greedy process, we keep the estimated coverage of each undeleted tasks above the minimum threshold. Thus, given $A \subseteq B \subseteq V_{f(1)}$, $(w_u, t_k) \in V_{f(1)} \setminus B$, we can see that when removing a single element (w_u, t_k) rom a larger set, the difference in decrement value increases, which is formally presented as follows:

$$h(B) - h(B \cup \{(w_u, t_k)\}) \geq h(A) - h(A \cup \{(w_u, t_k)\}) \quad (13)$$

The above inequality Eq. (13) is equal to

$$h(B \cup \{(w_u, t_k)\}) - h(B) \leq h(A \cup \{(w_u, t_k)\}) - h(A)$$

Therefore, according to the definition of submodular function [32], $h(V_r)$ is a submodular function. □

Proof 2. *OP-1 does not Affect the Optimality of the Original Problem*. We assume that there exists a set of task-worker pairs V^* which is an optimal allocation plan, task t_x is assigned to a certain set of workers in V^*, while t_x has been deleted by MTasker in OP-1. According to the temporal-spatial coverage estimation before OP-1 is performed, we can see that $\frac{|Covered(t_x, V_f)|}{|S \times C|} < MT(t_x)$. From the definition of overall utility (see Eq. (1)), we can see that the following equations are satisfied (we denote $V_x = \{(w_u, t_x) | (w_u, t_x) \in V^*\}$

$$\sum_{k=1}^{k=r} u_k(V^*) * wt_k = \sum_{k=1}^{k=r} u_k(V^* \setminus V_x) * wt_k. \quad (14)$$

Therefore, another different set of task-worker pairs $V^* \setminus V_x$ is also an optimal allocation plan. So that we can make the conclusion that OP-1 does not affect the optimality of the original problem. □

Proof 3. *OP-2 does not affect the Optimality of the Original Problem*. We assume that there exists a set of task-worker pairs V^* which is an optimal allocation plan, in which a task-worker pair $\{(w_u, t_k)\} \nsubseteq V^*$ *and* $\xi(w_u) \leq L_u$. We can see that another set of task-worker pairs denoted as $V^* \cup \{(w_u, t_k)\}$ is also a feasible solution, because the pre-condition $\xi(w_u) \leq L_u$ guarantees that the number of assigned tasks of w_u is not more than the maximum constraint. Meanwhile, we can also see that the following equations are satisfied

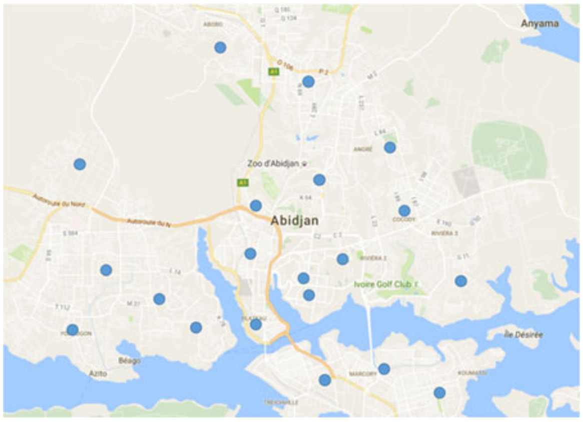


Fig. 4. Target sensing region used in the experiment (blue points represent cell towers).

$$\sum_{k=1}^{k=r} u_k(V^* \cup \{(w_u, t_k)\}) * wt_k = \sum_{k=1}^{k=r} u_k(V^*) * wt_k. \quad (15)$$

Therefore, another different set of task-worker pairs $V^* \cup \{(w_u, t_k)\}$ is the optimal allocation plan. So that we can make the conclusion that *OP-2 does not affect the optimality of the original problem*. □

5 Experimental Evaluation

In this section, we evaluate the effectiveness of MTasker using a real-world mobility trace dataset. We first introduce the dataset generation, experimental setup and baseline methods. Then, the evaluation results of MTasker and other baselines are demonstrated and compared in terms of overall utility and running time. Finally, we give detailed analysis for different approaches.

5.1 Datasets, Settings, and Baseline

5.1.1 *Datasets Description*

The dataset we used in evaluation is the D4D dataset [35], which contains two types of data records in Ivory Coast. One contains the information about cell towers, including tower id, latitude and longitude. The other one contains 50,000 users phone call records. We select users randomly every two weeks with anonymized ids and totally 10 two-week periods of records are stored in the dataset. In each two-week period, our experiment uses the first-week data records for multi-task allocation algorithm execution to obtain an allocation plan (i.e., a set of task-worker pairs), then we evaluate the overall utility of multiple tasks achieved by selected task-worker pairs using the second-week records. Specifically, we extracted records of the downtown area (20 cell towers with 3,500 mobile users), as shown in Fig. 4. We further assume that each task lasts for five days from Monday to Friday in a week, with 10 cycles every working day from 8:00 to 18:00 (one hour per cycle). Thus the total period consists of 50 sensing cycles.

5.1.2 *Experimental Setup*

First, the multi-task allocation problem we defined in this paper can be abstracted as a resource competition problem, in which the competitors are different tasks and resources are a shared pool of mobile users. There are three factors

TABLE 2
Parameter Settings

PARAMETERS	SETTINGS
r	3, 5, 10, 20, 30, 40, 50
l	500, 1000, 1500, 2000, 2500, 3000, 3500
μ	$\mu = 2, 4, 6, 8, 10$
wt_k	Randomly generated from (0,1) while ensuring the sum of different tasks is 1
$MT(t_k)$	Randomly generated from $(g, 1)$. $g = 0.4, 0.5, 0.6, 0.7, 0.8.$

(i.e., number of tasks r, number of workers l and maximum number of assigned tasks for each worker) which have direct influence on the resource competition. Thus, we need to evaluate the performance of different methods by varying these factors. We assume that the maximum number of assigned tasks for each worker follows Gaussian distribution with mean value μ and standard deviation σ . Second, for a fixed setting of the above factors, we randomly generate other parameters (e.g., the weights and minimum coverage threshold of each task, feasibility for task-worker pairs, etc.) from a reasonable range and conduct multiple rounds of experiments, in which the average performance is calculated as the final evaluation result. A naïve way to generate the minimum quality threshold of each sensing task from (0,1). However, this may not be reasonable, because it is not clear that what kind of task corresponds to the case that the threshold is very low (say 0.1). Thus, we randomly generate the minimum thresholds from $(g, 1)$ (g is a constant which satisfies $0 < g < 1$), and evaluate the performance of different approaches by varying g. Table 2 summarizes the parameter settings in our experiments. We carried out experiments using a laptop computer with an Intel Core i7-4710HQ Quad-Core CPU and 16 GB memory. MTasker and other baseline methods were implemented with the Java SE platform on a Java HotSpotTM 64-Bit Server.

5.1.3 Baseline Methods

We provide the following baseline task allocation methods for comparative studies.

- *Random Allocation (RA)*—This method randomly assigns tasks to each worker based on maximum number of assigned tasks and sensor availability. Since the order of workers may have an impact on the experimental results, we generate the orders randomly and repeat this random multi-task allocation process for 20 times. The highest overall utility is taken as the final result of RA.
- *Naive Ascent Greedy (Naïve-AG)*—The algorithm first picks the single task-worker pair with the maximal overall utility increase among all task-worker pairs and adds it to the set $V_f^{'}$. It then selects another task-worker pair with the maximal overall utility increase when combined with the selected pairs and adds it to $V_f^{'}$. Naïve-AG keeps selecting and adding new pairs until the number of assigned tasks of every user reaches the limit. The overall utility increase of adding a certain pair (w_u, t_k) is calculated based on $\sum_{k=1}^{k=r} u_k (V_f^{'} \cup \{(w_u, t_k)\}) * wt_k - \sum_{k=1}^{k=r} u_k(V_f^{'}) * wt_k$. However, because of the newly introduced task-specific minimum coverage threshold, it is likely that none of the task-worker pairs can lead to an overall utility increase when adding into V_f^* until one task meets the minimum coverage requirement. Thus, Naïve-AG randomly selects task-worker pairs in these cases.
- *Relaxed-Utility-Based Ascent Greedy (RU-AG)*—This task allocation process is almost the same as Naive-AG except for the utility function used when selecting task-worker pairs. When determining which task-worker pair should be selected, RU-AG uses a relaxed utility function, $\sum_{k=1}^{k=r} u_k^{'}(V_f^{'} \cup \{(w_u, t_k)\}) * wt_k - \sum_{k=1}^{k=r} u_k^{'}(V_f^{'}) * wt_k$, where $u_k^{'}(V_f^{'}) = \frac{|Covered(t_k, V_f)|}{|S \times C|}$ without considering its minimum coverage threshold.

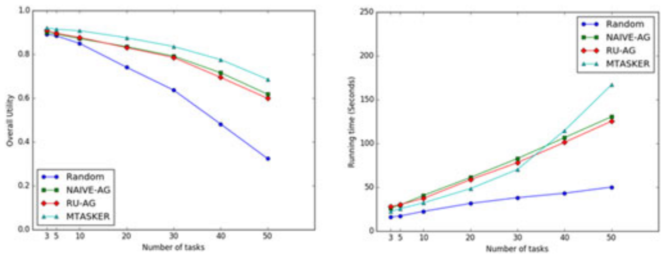


Fig. 5. Performance comparison for various number of tasks (left: overall utility and right: running time).

5.2 Experimental Results

In this subsection, we evaluate the performance of MTasker and other baseline methods by varying the following key parameters. Due to the space limitation, we cannot show the results for all possible parameter combinations. Instead, when varying one parameter, others are set as a fixed value.

5.2.1 Different Number of Tasks

In Fig. 5, we present the performance comparison on the overall utility and running time among MTasker and other baseline methods under different number of tasks, where we fix the number of workers as 2,000, mean value μ as 6 and g as 0.7.

From Fig. 5 (left), we can see that the overall utility decreases with the increasing number of tasks for all methods, because the competition for the limited resources among different tasks becomes more intense. MTasker outperforms RA, Naïve-AG and RU-AG by achieving higher overall utility for various number of tasks, respectively, and the advantage becomes more significant as the number of tasks increases. We can also see that the significance is not that obvious when the number of tasks is small (e.g., with 3 tasks). This is because when the number of tasks is small, the resource competition on a fixed group of workers is not that intense, so that coverage of each task can be high even by leveraging naive algorithms. However, the objective of this paper is to tackle the task allocation problem on open and large-scale multi-task MCS platforms. In practical application scenarios, such platforms must have the capability to support a large number of MCS tasks. Therefore, compared to other baseline methods, MTasker can better support such practical scenarios.

Fig. 5 (right) reports the running time of different methods. All results are measured on reasonably efficient

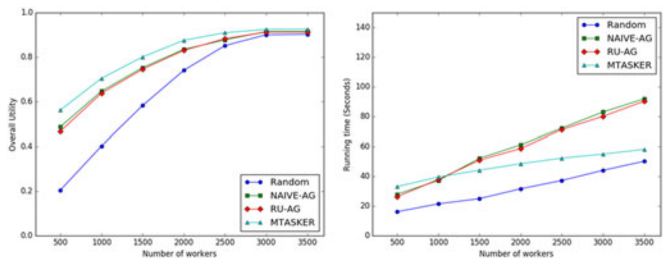


Fig. 6. Performance comparison for various number of workers (left: overall utility, right: running time).

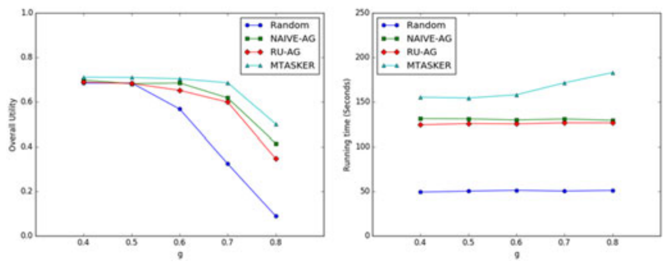


Fig. 8. Performance comparison for various g (left: overall utility, right: running time).

implementation of the various algorithms. Obviously, the fastest algorithm is RA. The running time of MTasker is longer than Naïve-AG and RU-AG in some settings while shorter in some other settings. Although MTasker needs longer running time in some settings than other baselines, its running time is less than five minutes for various number of tasks. Since the algorithm is executed offline, the computation time is acceptable. Besides, in this experiment, the MTasker runs on a laptop computer. To implement a real-world MCS system, shorter computation time can be achieved by using parallel algorithms or deploying MTasker on a more powerful commercial server.

5.2.2 Different Number of Workers

In Fig. 6, we present the performance comparison on the overall utility and running time among different methods under different number of workers, where we fix the number of tasks as 20, mean value μ as 6 and value of g as 0.7.

From Fig. 6 (left), we can see that the overall utility increases with the number of workers for all methods. This is because with increasing number of workers, the shared resources become more abundant. MTasker outperforms RA, Naïve-AG and RU-AG by achieving higher overall utility for various number of workers. We can see that although MTasker outperforms other baseline methods in all settings, the significance become less obvious with the increase in number of workers, which can be cognized from the following two aspects. On the one hand, the number of available workers can be small when the platform has just been developed, because mobile users in the city do not know the existence of the platform. In this case, MTasker outperforms other baselines significantly in the overall utility. On the other hand, even if there are an adequate number of available workers, MTasker can recruit fewer workers when achieving almost the same overall utility. This is beneficial because recruiting fewer workers can reduce the total incentive cost of the platform.

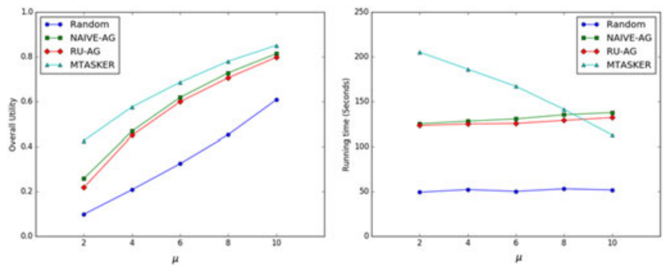


Fig. 7. Performance comparison for various μ (left: overall utility, right: running time).

Fig. 6 (right) reports the running time of different algorithms, and we can see that the running time MTasker needs is longer than other baselines methods in some settings while shorter in some other settings. However, the running time of MTasker is less than 5 minutes for various number of workers, which is acceptable as it is executed offline.

5.2.3 Different Values of μ

When varying the parameter μ, we fix the number of tasks as 50, the number of workers as 2,000 and value of g as 0.7. Fig. 7 (left) shows the overall utility for different values of μ. From the figure, we can see that under various settings of mean value μ, MTasker outperforms RA, Naïve-AG and RU-AG by achieving higher overall utility. As Fig. 7 (right) shows, the running time that MTasker needs is longer than other baseline methods in some settings while shorter in other settings. In addition, we can observe that with the increase of μ, the running time of MTasker decreases while the running time of other methods increases. This phenomenon can be explained by re-checking the execution process of different algorithms. Both Naïve-AG and RU-AG adopt an ascent greedy algorithm, which select task-worker pairs incrementally until the number of assigned tasks for each worker reach to the maximum constraint. In contrast, MTasker searches for the quasi-optimal allocation based on a greedy descent algorithm, which removes task-worker pairs from the full set until the number of assigned tasks for each worker meets the requirement. Therefore, as the value of μ increases, the number of iterations that Naïve-AG and RU-AG need increases, while fewer iterations are needed by MTasker.

5.2.4 Different Values of g

In this experiment, we randomly generate the minimum thresholds from $(g, 1)(0 < g < 1)$. As we cannot list what kind of task corresponds to the case that the threshold is very low, we vary g as 0.4, 0.5, 0.6, 0.7, 0.8. When varying the parameter g, we fix the number of tasks as 50 and the number of workers as 2,000, and fix the mean value μ as 6. Fig. 8 shows how the changes in quality requirement affect the overall quality. It demonstrates that MTasker outperforms other baseline methods in terms of obtained overall utility in different settings, and the advantage becomes more significantly with the increase of g. Besides, the running time of MTasker increases slightly with the increase of g.

5.3 Detailed Analysis

5.3.1 MTasker vs. Naïve-AG

Fig. 9 shows the number of assigned workers for tasks under a specific setting (10 tasks, 2,000 workers, $\mu = 6$ and

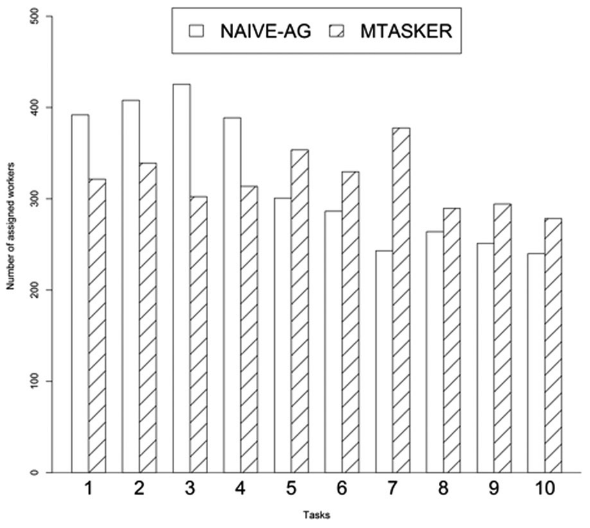


Fig. 9. Number of assigned workers for each task (tasks are sorted in increasing order of minimum threshold).

a given settings of minimum thresholds). In Fig. 9, the tasks are ordered based on its minimum temporal-spatial thresholds (task #1 with lowest threshold while task #10 with highest threshold). We can see that for Naïve-AG the tasks with lower minimum coverage thresholds tend to be assigned to more workers. During the execution process of Naïve-AG, with the addition of task-worker pairs, tasks with lower thresholds would reach their thresholds earlier, so that the algorithm tends to allocate more workers to them. However, this characteristic may count against the overall utility optimization. For example, when the tasks with lower thresholds are less important (i.e., with smaller weights), more limited resources are assigned to tasks with smaller weights, and tasks with larger weights receive fewer assigned workers, which is bad for the overall utility optimization. In contrast, from Fig. 9 we can also see that the relationship between the minimum temporal-spatial coverage threshold and number of assigned workers is not obvious for MTasker. In summary, MTasker outperforms the Naïve-AG by reducing the impact of minimum coverage threshold on resource allocation.

5.3.2 *MTasker vs. Ru-AG*

To clarify the reason why MTasker outperforms RU-AG, we investigate the obtained temporal-spatial coverage of each single task. Specifically, all tasks are divided into three categories: (1) C_1: tasks without assigned workers; (2) C_2: tasks with some assigned workers but the achieved coverage is lower than the thresholds; (3) C_3: tasks whose achieved coverage is not lower than the thresholds. In this experiment, we evaluate the distribution of the above three categories

TABLE 3
The Distribution for Three Types of Tasks: MTasker vs. RU-AG (Average Number of Tasks for Various Settings of Weights and Minimum Thresholds)

	MTasker	RU-AG
C_1	12.1	0
C_2	0.4	15.4
C_3	37.5	34.6

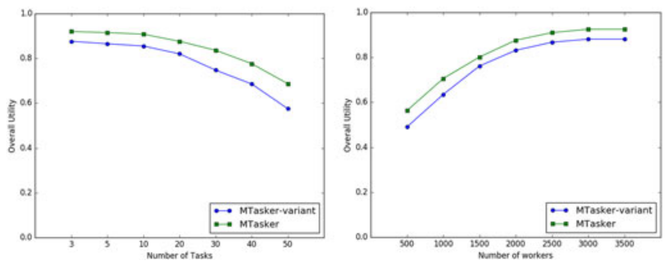


Fig. 10. Overall utility comparison between MTasker and its variant.

under different number of tasks and workers. Due to the space limitation, Table 3 shows the results when fixing the number of workers as 2,000 and the number of tasks as 50, which is the average performance when generating different settings of weights and thresholds.

From Table 3 we can see that RU-AG assign 50 tasks with some workers. However, there are only averagely 34.6 tasks in C_3(with quality-assurance), while 15.4 tasks fall into C_2. In contrast, MTasker tries its best to assign workers to the tasks with quality-assurance while assigning no workers to other discarded tasks (only with on average 0.4 tasks in C_2). Intuitively, tasks in C_2 are bad for the multi-task allocation optimization, because resources are consumed yet zero-contribution is made to the overall utility. Therefore, the advantage of MTasker, compared with RU-AG, lies in the fact that it reduces the wasted resources by allocating nearly all resources to the tasks whose minimum coverage can be assured. We also note that MTasker cannot guarantee that the number of tasks in C_2 is zero, because the mobility prediction estimation is not 100 parcent accurate. Thus, we plan to improve mobility prediction methods in future work, which will help to further improve the overall utility.

5.3.3 *Effectiveness of Mobility Prediction*

In order to explore the impact of the mobility prediction on the overall system performance, we implement a variant of MTasker which uses a more naïve mobility prediction method (called MTasker-variant). This variant version simply uses the statistical result of a worker's location records to measure the probability that she/he will pass through a subarea. We compare the overall utility of MTasker with this variant when varying the number of tasks and workers, which is shown in Fig. 10 (left) and 10 (right) (fixing the value of as 0.7 and mean value μ as 6). The results show that the mobility prediction does have impact on the overall system performance, and using more sophisticated prediction algorithm can help improve the achieved overall utility.

Besides, in addition to real-world dataset, one of the commonly used mobility model for simulations is "Time-variant Community Mobility Model" (TVCM) [36]. This is helpful in verifying the mobility prediction component's effectiveness. With TVCM, we can simulate the frequently visited communities (areas that a node visits frequently) and different time periods in which the node periodically re-appears at the same location. Thus, we generate the mobility traces of the workers based on the TVCM, and re-test the performance of MTasker with the baseline methods when varying the movement speed and average range of frequently visited communities in TVCM, which is shown in Fig. 11 (left) and 11 (right) (fixing the number of tasks as 20, the number

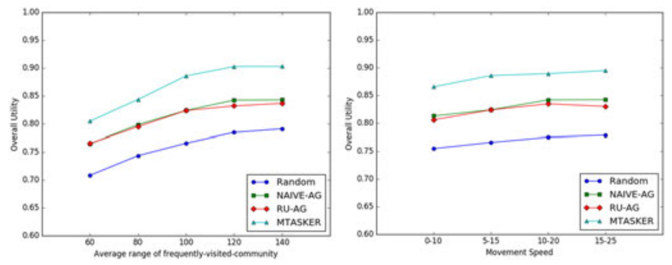


Fig. 11. Overall utility comparison on simulated mobility dataset based on TVCM.

of workers as 1,000, value of as 0.7, and mean value μ as 6). The results show that MTasker still outperforms other baselines on the TVCM-simulated mobility dataset.

6 Discussion

This section discusses issues that are not reported or addressed in this work due to space and time constraints, which can be added to our future work.

- *Various Factors and Constraints.* We consider two worker-side constraints when formulating our multi-task allocation problem. Other types of factors may need to be considered in a real-world multi-task MCS system, such as the budget of MCS platforms, the interests of workers, etc. Besides, this paper assumes that sensing subareas and cycles are the same across different tasks. However, we may encounter more complex situations, where subareas and cycles are different for multiple tasks. Therefore, we plan to improve the practicality of our proposed framework by considering other constraints in our future work.
- *Sensing Quality Metrics.* In this paper, we utilize the temporal-spatial coverage metric to measure the sensing quality. Similar to many state-of-the-art research works for MCS [10], [27], [28], we assume that a temporal-spatial cell is covered if the system obtains at least one sensor reading. However, [30] points out that we may need more sensor readings in a temporal-spatial cell to guarantee the reliability of sensing data, and more complex sensing quality metrics need to be considered. In our future work, we intend to evaluate the effectiveness of MTasker with different sensing quality metrics.
- *Mobility Prediction.* The experimental result reveals that the mobility prediction does affect the overall system performance, so that it is worth designing more sophisticated prediction algorithm to further improve MTasker. For example, inspired by [37], [38], we can leverage the mobility correlation among different workers to infer some missing traces in our future work, which may help improve the prediction accuracy.
- *Task Dependency.* This paper only considers the competitive relation among tasks. That is, if a sensing resource (workers) is allocated to some tasks, other tasks cannot utilize it. However, we can take into account more complicated situations, where sensing results for a task can be utilized for another task. For example, some tasks can share the same type of sensing data, or the sensing data among some tasks are co-related.
- *Subareas and Privacy Concerns.* Due to the characteristics of the D4D dataset, we can only measure the temporal-spatial coverage and evaluate our approach at the cell tower level. Since the algorithm of MTasker is general, if more fine-grained user mobility traces can be collected, we could support the multi-task MCS system at finer-grained levels (e.g., subareas with 100 m*100 m). However, it also leads to location privacy concerns. The current version of MTasker uses historical trajectory records to establish predictive models. One way to reduce the privacy threats is to only provide predictive models, rather than raw trace records, to MTasker, as supplied by the mobile operators. Another alternative is that the MTaskers client side on the users device can collect raw location data, but only upload predictive models for the multi-task allocation algorithm.
- *Achievable Sensing Quality.* In the theoretical analysis, we can see that the task allocation algorithm of MTasker is quasi-optimal in terms of the overall utility. However, in the current version, we cannot find theoretical relation between the individual tasks exact achievable quality and the number of tasks/workers, which can be added into the future work lists.
- *Large-Scale User Study.* In this work, although we evaluate the effectiveness of our algorithm based on an open dataset for real-world mobility traces, some parameters about tasks or workers (e.g., number of tasks, the tasks weights, and average number of assigned tasks, etc.) are simulated with certain assumptions. We are now cooperating with the local government to develop a city-scale MCS platform for ordinary citizens to report information (e.g., traffic jams, missing Manhole cover, etc.). We intend to test the core algorithm of MTasker in the platform to identify more practical issues for further improvement.

7 Conclusion

In this paper, we studied a novel multi-task allocation problem by introducing task-specific minimum sensing quality thresholds, with the objective of assigning appropriate sets of tasks to workers for maximizing the overall system utility. Our problem also considers the maximum number of tasks allowed for each worker and the sensor availability of each mobile device. To solve the above problem, we proposed the MTasker framework, which adopts a descent greedy approach to select a quasi-optimal set of task-worker pairs. It first pseudo-allocates task-worker pairs and checks the validity of some pairs, based on which the original overall utility maximization problem is transformed into the problem of removing a set of task-worker pairs from the full set to minimize the overall utility reduction. MTasker then utilizes an iterative greedy process to optimize this transformed problem based on the theory of submodular function maximization. Extensive evaluations based on real-world mobility traces demonstrate that MTasker outperforms the baseline methods under various settings. As for future work, we will consider other factors that may affect multi-task allocation in MCS systems. More sophisticated optimization methods and theoretical foundations will also be explored.

ACKNOWLEDGMENTS

This work was supported by NSFC under Grant No. 61702017 and the National Key R&D Program of China under Grant 2017YFB1002000.

REFERENCES

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

[2] B. Guo, et al., "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surveys*, vol. 48, no. 1, 2015, Art. no. 7.

[3] Y. Zheng, F. Liu, and H. Hsieh, "U-Air: When urban air quality inference meets big data," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 1436–1444.

[4] V. Coric and M. Gruteser, "Crowdsensing maps of on-street parking spaces," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, 2013, pp. 115–122.

[5] W. Wu, B. Guo, and Z. Yu, "Crowd sensing based urban noise map and temporal-spatial feature analysis," *J. Comput.-Aided Design Comput. Graphics*,vol. 26, no. 4, pp. 638–643, 2014.

[6] Campaignr. [Online]. Available: http://research.cens.ucla.edu/urban/

[7] M. Ra, B. Liu, T. F. L. Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *Proc. 10th Int. Conference Mobile Syst., Appl. Services*, 2012, pp. 337–350.

[8] A. Kittur, et al., "The future of crowd work," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, 2013, pp. 1301–1318.

[9] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proc. ACM 2012 Conf. Comput. Supported Cooperative Work*, 2012, pp. 1003–1012.

[10] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proc. ACM Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 703–714.

[11] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "CrowdTasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2015, pp. 55–62.

[12] M. Zhang, P. Yang, C. Tian, and S. Tang, "Quality-aware sensing coverage in budget constrained mobile crowdsensing networks,"*IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7698–7707, Sep. 2015.

[13] A. Singla and A. Krause, "Incentives for privacy tradeoff in community sensing," in *Proc. 1st AAAI Conf. Human Comput. Crowdsourcing*, 2013, pp. 165–173.

[14] Z. Song, B. Zhang, C. H. Liu, A. Vasilakos, V. J. Ma, and W. Wang, "QoI-aware energy-efficient participant selection," in *Proc. 11th Annu. IEEE Int. Conf. Sens., Commun., Netw.*, Jun. 2014, pp. 248–256.

[15] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "Activecrowd: A framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 3, pp. 392–403, Jun. 2017.

[16] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proc., IEEE Conf. Comput. Commun.*, 2015, pp. 2254–2262.

[17] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazons mechanical turk a new source of inexpensive, yet high-quality, data?" *Perspectives Psychological Sci.*, vol. 6 pp. 3–5, 2011.

[18] CrowdFlower. [Online]. Available: https://www.crowdflower.com/

[19] Samasource. [Online]. Available: http://www.samasource.org/

[20] S. B. Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das, "Task assignment optimization in knowledge-intensive crowdsourcing," *VLDB J.*, vol. 24, no. 4, pp. 467–491, 2015.

[21] S. Assadi, J. Hsu, and S. Jabbari, "Online assignment of heterogeneous tasks in crowdsourcing markets," in *Proc. Third AAAI Conf. Human Comput. Crowdsourcing*, 2015, pp. 12–21.

[22] G. Goel, A. Nikzad, and A. Singla, "Mechanism design for crowdsourcing markets with heterogeneous tasks," in *Proc. Second AAAI Conf. Human Comput. Crowdsourcing*, 2014, pp. 77–86.

[23] S. Hachem, A. Pathak, and V. Issarny, "Probabilistic registration for large-scale mobile participatory sensing," in *Proc. IEEE Int. Conference Pervasive Comput. Commun.*, 2013, vol. 18, pp. 132–140.

[24] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: Platform for remote sensing using smartphones," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Services*, 2010, pp. 63–76.

[25] J. Hicks, et al., "AndWellness: An open mobile system for activity and experience sampling," in *Proc. Wireless Health*, 2010, pp. 34–43.

[26] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using context annotated mobility profiles to recruit data collectors in participatory sensing," in *Proc. Int. Symp. Location Context Awareness*, 2009, pp. 52–69.

[27] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Proc. Pervasive*, 2010, pp. 138–155.

[28] G. Cardone, et al., "Fostering participaction in smart cities: A geo-social crowdsensing platform," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 112–119, Jun. 2013.

[29] M. Xiao, J. Wu, L. Huang, Y. Wang, and C. Liu, "Multi-task assignment for CrowdSensing in mobile social networks," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2227–2235.

[30] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, and D. Zhang, "TaskMe: Multi-task allocation in mobile crowd sensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 403–414.

[31] Z. Song, C. H. Liu, J. Wu, J. Ma, and W. Wang, "QoI-aware multitask-oriented dynamic participant selection with budget constraints," *IEEE Trans. Veh. Technol.*, vol. 63, no. 9, pp. 4618–4632, Nov. 2014.

[32] J. Wang, et al., "Fine-grained multi-task allocation for participatory sensing with a shared budget," *IEEE Internet Things J.*, vol. 3, pp. 1395–1405, Dec. 2016.

[33] J. Wang, Y. Wang, D. Zhang, F. Wang, Y. He, and L. Ma, "PSAllocator: Multi-task allocation for participatory sensing with sensing capability constraints," in *Proc. ACM Conf. Comput.-Supported Cooperative Work Soc. Comput.*, 2017, 1139–1151.

[34] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches Hard Problems*, vol. 3, no. 19, pp. 71–104, 2012.

[35] V. D. Blondel, et al., "Data for development: The d4d challenge on mobile phone data," arXiv preprint arXiv:1210.0137, 2012.

[36] W.-J. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling time-variant user mobility in wireless mobile networks," in*Proc. 26th IEEE Int. Conf. Comput. Commun.*, 2007, pp. 758–766

[37] B. Hawelka, I. Sitko, P. Kazakopoulos, and E. Beinat, "Collective prediction of individual mobility traces for users with short data history," *PloS One*, vol. 12, no. 1, 2017, Art. no. e0170907.

[38] L. Kong, L. He, X.-Y. Liu, Y. Gu, M.-Y. Wu, and X. Liu, "Privacy-preserving compressive sensing for crowdsensing based trajectory recovery," in *IEEE 35th Int. Conf. Distrib. Comput. Syst.*, 2015, 31–40.

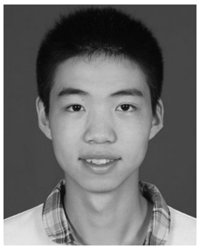
[39] J. Wang, Y. Wang, L. Wang, and Y. He, "GP-selector: A generic participant selection framework for mobile crowdsourcing systems," in *Proc. World Wide Web*, 2007, pp. 1–24.

[40] J. Wang, Y. Wang, D. Zhang, and S. Helal, "Energy saving techniques in mobile crowd sensing: Current state and future opportunities," *IEEE Commun. Mag.*, (accepted, to appear in 2018).

Jiangtao Wang received the PhD degree from Peking University, Beijing, China, in 2015. He is currently an assistant professor in the Institute of Software, School of Electronics Engineering and Computer Science, Peking University. His research interest includes collaborative sensing, mobile computing, and ubiquitous computing.

Yasha Wang received the PhD degree from Northeastern University, Shenyang, China, in 2003. He is a professor and an associate director of the National Research & Engineering Center of Software Engineering at Peking University, China. His research interest includes urban data analytics, ubiquitous computing, software reuse, and online software development environment. He has published more than 50 papers in prestigious conferences and journals, such as ICWS, UbiComp, ICSP, etc. As a technical leader and manager, he has accomplished several key national projects on software engineering and smart cities. Cooperating with major smart-city solution providing companies, his research work has been adopted in more than 20 cities in China.

Daqing Zhang received the PhD degree from the University of Rome La Sapienza, Italy, in 1996. He is a professor with Peking University, China, and Télécom SudParis, France. His research interests include context-aware computing, urban computing, mobile computing, and so on. He served as the general or program chair for more than 10 international conferences. He is an associate editor for the *ACM Transactions on Intelligent Systems and Technology*, the *IEEE Transactions on Big Data*, and others.



Feng Wang is working toward an undergraduate degree in the School of Electronic Engineering and Computer Science, Peking University, China. His research interest is mobile crowd sensing.

Haoyi Xiong received the PhD degree with highest honors in computer science from Telecom SudParis and the University of Paris VI, France, in 2015. He is currently an assistant professor in the Department of Computer Science, Missouri University of Science & Technology, in Rolla, Missouri (previously known as University of Missouri at Rolla). From July 2015 to Aug. 2016, he was a postdoctoral research associate in the Department of Systems and Information Engineering, University of Virginia, Charlottesville, Virginia. His research interests include ubiquitous computing, cyber-human systems, and large-scale optimization & decision making. He received the Best Paper Award from IEEE UIC 2012 and the Outstanding PhD thesis Runner-up Award from CNRS SAMOVAR 2015. He is a member of the IEEE.

Chao Chen received the PhD degree from Pierre and Marie Curie University, Paris, France, in 2014. He is an associate professor in the College of Computer Science, Chongqing University, Chongqing, China. His research interests include pervasive computing, urban logistics, data mining from large-scale taxi data, and big data analytics for smart cities.

Qin Lv received the PhD degree in computer science from Princeton University, in 2006. She is an associate professor in the Department of Computer Science, University of Colorado Boulder. Her main research interests include data-driven scientific discovery and ubiquitous computing. Her research spans the areas of multi-modal data fusion, spatial-temporal data analysis, anomaly detection, mobile computing, social networks, recommender systems, and data management. Her research is interdisciplinary in nature and interacts closely with a variety of application domains including environmental science, Earth sciences, renewable and sustainable energy, materials science, as well as the information needs in peoples daily lives. She is an associate editor of the Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), and has served on the technical program committee and organizing committee of many conferences. Her work has more than 4,000 citation.

Zhaopeng Qiu is working toward an undergraduate degree in the School of Electronic Engineering and Computer Science, Peking University, China. His research interest is data analysis and NLP.