Allocating Heterogeneous Tasks in Participatory Sensing with Diverse Participant-Side Factors

Jiangtao Wang, Feng Wang, Yasha Wang, Daqing Zhang, Brian Lim, and Leye Wang

Abstract—This paper proposes a novel task allocation framework, PSTasker, for participatory sensing (PS), which aims to maximize the overall system utility on PS platform by coordinating the allocation of multiple tasks. While existing studies mainly optimize the task allocation from the perspective of the task organizer (e.g., maximizing coverage or minimizing incentive cost), PSTasker further considers diverse factors on the participants' side, including user work bandwidth, user availability, devices' sensor configuration, task completion likelihood, and mobility pattern. Furthermore, by considering the heterogeneity in three dimensions (i.e., task, time and space), it adopts a novel model to measure task sensing quality and overall system utility. In PSTasker, it first calculates the utlity of a given task allocation plan by jointly fusing different participant-side factors into one unified estimation function, and then employs an iterative greedy process to optimize the task allocation. Extensive evaluations based on real-world mobility traces demonstrate that PSTasker outperforms the baseline methods under various settings.

Index Terms—Participatory sensing, mobile crowd sensing, task allocation, participant-side factors

1 Introduction

PARTICIPATORY sensing (PS) or mobile crowd sensing [1], [2] is a novel paradigm for monitoring the urban landscape. In a PS system, task organizers publish sensing tasks with certain specifications (e.g., sensing target area and duration), while ordinary citizens (called participants) collect sensing data from their surrounding environment with their mobile devices, and report to the cloud server. The data contributed from multiple participants can be aggregated to build a spatio-temporal view of the phenomenon of interest in a city (e.g., pollutant distribution monitoring, noise map construction and traffic congestion detection, etc.).

Many PS platforms such as Campaignr [3] and Medusa [4] have been developed. In order to reduce the burden of participants in finding appropriate sensing tasks on the PS platform, automatic task allocation has been studied extensively in recent years [11], [12], [14], [15], [16], but these approaches are mostly single-task-oriented. This assumes that tasks are independent and there are always sensing resources (participants or mobile devices) available for each task. Based on the above assumptions, their goal is to select an optimal subset of mobile users from a large user pool to complete the sensing task with certain optimization goals (e.g., spatio-temporal coverage and incentive cost).

However, with more and more applications leveraging PS paradigm, sensing tasks are not independent on a multi-task platform, because they compete with each other in a shared and limited resource pool (i.e., the same set of participants). To this end, some recent studies (e.g., [20], [22], [23], [24]) have started to focus on multi-task allocation, where the interdependency of multiple tasks is considered. Although the corresponding multi-task allocation frameworks or approaches are proven to be effective in solving their formulated problems, the following limitations still exist.

First, existing studies do not fully consider the heterogeneity when modeling and quantifying overall utility of multiple PS tasks. For PS tasks, a number of sensor data samples are needed to guarantee the reliability of the collected data [11]. In a multi-task PS platform, we observe that in terms of the required number of data samples, there exist heterogeneity among tasks, sensing cycles and subareas. First of all, different sensing tasks require different numbers of data samples to guarantee a certain level of reliability. For example, the number of data samples needed for a noise level sensing task is larger than that for a air quality monitoring task, because the noise level may be very diverse even among nearby locations, while the distribution of air quality is much more uniform. Furthermore, even for the same task, the required number of data samples in a different time (sensing cycles) and space (subareas) is also different. By taking the traffic detection task as an example, in terms of space, it requires more data samples for the subareas in downtown than those in the suburb. In terms of time, the task will also require more samples in rush hours than at night.

Second, existing studies mainly optimize the PS task allocation from the perspective of task organizers (e.g., task's spatio-temporal coverage, the number of completed tasks, budget constraints, etc.), while neglecting participant-side factors. For example, both [23] and [24] aim at optimizing the overall utility of multiple tasks while keeping a total budget constraint for the platform, and the authors in [20], [31] propose frameworks to minimize the incentive cost while ensuring the completion of required sensing tasks. In these research works, participant-side factors, such as participants' bandwidth, availability, devices' sensor configuration and task completion likelihood, are rarely taken into

- *Jiangtao Wang, Feng Wang, Yasha Wang, Daqing Zhang are with the Peking University, Beijing, China.*
E-mail: jiangtaowang@pku.edu.cn;
- *Brian Lim is with the National University of Singapore, Singapore.*
- *Leye Wang is with the Hong Kong University of Science and Technology, Hongkong, China*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2869387, IEEE Transactions on Mobile Computing

consideration. They commonly assume that participants are able to perform an unlimited number of tasks, or assume that participants complete tasks at any time and locations if the task is assigned, which does not conform to the practical application scenarios.

Considering the heterogeneity among tasks and the diverse participant-side factors, this paper defines a novel multi-task allocation problem. Specifically, the objective is to assign an appropriate set of tasks to each participant (i.e., selecting an appropriate set of task-and-participant pairs) to maximize the overall system utility, with consideration of the following participant-side factors.

- *Participant bandwidth.* Each participant is willing to be assigned with a limited number of tasks, as they do not want to be disturbed frequently. Thus, in this paper, we allow each participant to pre-define their maximum number of assigned tasks (called bandwidth).
- *Participant availablility.* Delivering PS tasks to participants brings intrusiveness and privacy risk to them. In fact, participants may not be willing to receive and complete during some periods of time or at some locations. For example, some may not want to be disturbed during working time, while others may reject the task when they are at home for location privacy concerns. Thus, in this paper, we allow each participant to explicitly define the spatio-temporal cells where they do not want to do tasks.
- *Sensor configuration.* The participant's mobile device might not possess the required sensors. Besides, even if having required sensors, the participants may also choose to actively disable some (e.g., those sensors collecting sensitive data). The sensor configuration information can be automatically captured by the PS platform when the participants registered their mobile devices.
- *Task completion likelihood.* State-of-the-art research works assume that the participants would complete each assigned task. However, the recent literature [37] reveals that this assumption may not be true and people usually complete the assigned tasks with a certain probability. In this paper, we assume that each participant has a task completion likelihood, which could be learned through his/her historical participation records.

The aforementioned overall utility maximization problem is NP-hard[1], so that it is impossible to use a brute-force enumeration approach. Thus, we require a feasible mechanism to optimize the multi-task allocation while ensuring that the computation complexity is under control.

The main contributions of this paper are:

(1) This paper investigates and formulates a novel multi-task allocation problem for PS. First, while most of the existing work mainly optimizes the task allocation from task organizers' perspective, we take into account several participant-side factors, including the bandwidth, availability, devices' sensor configuration and task completion likelihood. Second, by considering the heterogeneity among tasks and spatio-temporal cells, we adopt a novel model to measure task sensing quality and define the overall system utility.

(2) We convert the multi-task allocation problem into the representation of a bipartite graph and propose a multi-task allocation framework, called PSTasker. PSTasker first reduces the complexity of the original problem by assigning feasible tasks to underloaded participants. It then takes an iterative greedy process to optimize task assignments for overloaded participants. Specifically, it calculates the utility increase achieved by different task-and-participant pairs by jointly modeling diversifying participant-side factors in one unified function.

(3) We evaluate PSTasker extensively with real-world mobility traces and simulations. We verify that, under various settings, PSTasker outperforms baseline methods by achieving higher overall utility.

2 Related Works

2.1 Task Allocation in Online Crowdsourcing

Many online crowdsourcing platforms (e.g.,[25], [26], [27]) have been developed in recent years. Meanwhile, task allocation is also extensively studied for online crowdsourcing, where corresponding approaches [28], [29], [30], [34], [35] are proposed with different optimization goals and constraints. These approaches are designed for human intelligence tasks (e.g., image classification), which are completed online (e.g., through a desktop computer at home). On the contrary, the goal of PS tasks is to sense physical environment or phenomenon in cities (e.g., temperature, traffic status, and air quality). This is location-dependent and requires the participant's physical presence at some locations. Therefore, some key factors, such as the user's mobility, sensor configuration and heterogeneity among different tasks, should be taken into account in PSTasker. Thus, existing task allocation approach for online crowdsourcing cannot be directly adopted to solve our formulated problem.

2.2 Single Task Allocation in PS

There are two main groups of research in single task allocation for PS. A group of research aims at minimizing the cost while ensuring a certain degree of the data quality. Xiong et al. [9], [10] investigated how to assure the full coverage over a several cell towers with the minimum number of users. Zhang et al. [8] studied offline participant selection for probabilistic coverage, whose objective is to select a minimum number of participants to guarantee that a certain percentage (e.g., 85%) of coverage. Karaliopoulos et al. [15] studied the user recruitment problem, whose goal is to minimize the total incentive payment while ensuring each points-of-interest is visited by participants before the deadline. Hachem et al. [16] proposed a participant selection framework for participatory sensing, which reduces the number of selected participants by predicting mobile users' future locations in next time slot. Song et al. [14] proposed a QoI-aware and energy-efficient participant selection strategy, which aims at balancing the quality of Information and

[1] The NP-hardness of the problem is proved in the preliminary conference version of this work [33].

energy cost.In [17], [18], the authors leveraged the spatial or temporal correlations to reduce the number of participants, while meeting certain coverage quality constraints. In [19], the authors proposes a compressive sensing based approach to select a minimum number of participants while ensuring the sensing accuracy.

In contrast, another set of literature aims at maximizing the data quality of PS with certain constraints. In [5], [6] Reddy et al. investigated the research challenges of participant recruitment in PS and proposed a strategy to select a predefined number of participants so as to maximize the spatial coverage. In [11], [12], the authors attempt to maximize the coverage quality under an overall budget constraint. Cardone et al. [7] developed a platform, where a simple participant selection mechanism was proposed to maximize the spatial coverage of crowd sensing with a predefined number of participants. Singla et al. [13] designed an adaptive participant selection mechanism for maximizing spatial coverage with the limited total budget in community sensing with privacy preserving concerns.

These research works focused on single task allocation and did not consider the interdependency among multiple tasks. In contrast, PSTasker optimizes the task allocation of multiple concurrent PS tasks.

2.3 Multi-Task Allocation in PS

In recent years, researchers start to focus on multi-task allocation of PS [22], [24], [31], [32], [33]. In [22], the optimization goal is to minimize total cost while ensuring different levels of coverage for multiple tasks. In [24], the goal is to achieve the best quality satisfaction metrics for all tasks under the total budget constraints. Another work [31] aims at minimizing the overall sensing cost of mobile devices while completing all location-specific tasks. Yang et al. [32] proposed an approach to minimize the total latency of sensing tasks to ensure the quality of data. Authors in [20], [21] studied two cases for multi-task mobile crowd sensing (i.e., few participants many tasks, and many participants few tasks). Our previous conference paper [33] also formulated a multi-task allocation problem by taking the task coverage quality and participant's maximum bandwidth into account.

These works mainly optimize the multi-task allocation from the perspective of task organizer without fully taking into account participant-side factors. For example, most of them assume that participants are able to perform an unlimited number of tasks, or assume that participants complete tasks at any time and location, which does not conform to the practical application scenarios. On the contrary, PSTasker takes into account diverse participant-side factors, including the bandwidth, availability, devices' sensor configuration and task completion likelihood. In addition, by considering the heterogeneity in three dimensions (i.e., task, time and space), PSTasker measures task sensing quality and overall system utility more practically.

3 System Model and Problem Definition

In this section, we first display a running use case in subsection 3.1, which is used throughout the paper. Second, we model the data quality by considering heterogeneity in three dimensions, and further define the overall system utility in subsection 3.2. Third, we present the participant-side factors we consider in PSTasker in subsection 3.3. Finally, the optimization problem is formally formulated in subsection 3.4. The important notations are summarized in Table 1.

TABLE 1
Major notations used throughout this paper

Notation	Interpretation
$t_1, \ldots, t_k$	A set of tasks
$p_1, \ldots, p_H$	A set of participants
$MinT_x, MaxT_x$	Minimum and maximum number of samples for a subarea in a cycle of task x
$ES^x_{i,j}$	The effective number of samples for a specific task x on a subarea i and in a cycle j
$Cell\_DQI^x_{i,j}$	Data quality of a specific task x on a spatio-temporal cell (subarea i and cycle j)
$\delta^x_{i,j}$	The importance of each spatio-temporal cell
DQI^x	Data quality of a specific task x
$U(\Psi)$	PS system's overall utility achieved by task allocation plan Ψ
V_f	A set of task-and-participant pairs (task allocation instance set)
$\beta_{u,i,j}$	Representing if participant p_u allow the system to assign tasks in subarea i and cycle j
L_u	The bandwidth of participant p_u
$\tilde{r}_{u,i}$	the likelihood that p_u completes tasks in subarea i

Similar to [9], [10], [33], we also uses the cell tower as the sensing range of each subarea (i.e., sensing task) due to two reasons: 1) The cell tower IDs of mobile phones are accessible in call logs, thus using cell towers as subarea division metrics can illustrate the core idea of PSTasker; 2) For PS applications, such as urban air quality monitoring, covering a high percentage of cell towers ensures that most parts of the entire sensing region is measured, even though the granularity of cell tower may not be the best choice. However, the key algorithms of PSTasker are not restricted by how the subareas are divided, and using cell towers as subarea division metrics is just an example to illustrate the core idea of PSTasker.

3.1 Use Case Description

*To aid in understanding, we present a use case with ParSens, a PS-driven platform, which shares real-time location-dependent information in a city. 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, where the region covered by a cell tower is regarded as a subarea (see **Figure 1**). The entire sensing duration is divided into equal-length cycles (e.g. 2 hours per cycle). In ParSense there are multiple sensing tasks (e.g., air quality monitoring, traffic congestion detection, and noise map construction), which share the same subareas and cycles. Each task is published by an organizer with task-specific data quality requirement. There are a set of mobile users registered in ParSense as candidates, and they share their cell tower connection records to the platform. Each participant predefines the maximum number of assigned tasks (bandwidth) and availability in different spatial-temporal cells through ParSense's*

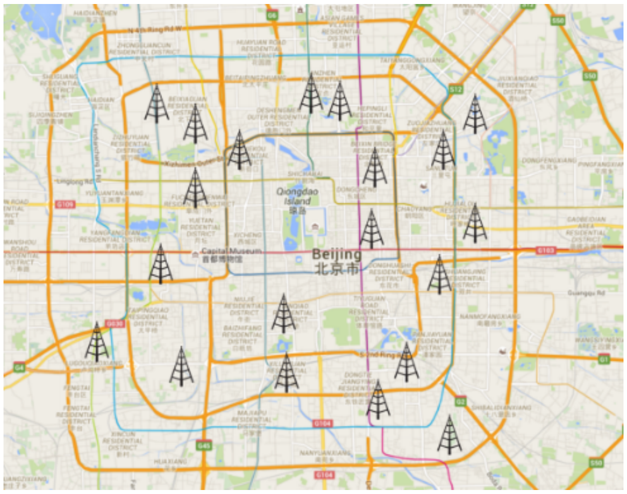


Fig. 1. Sensing area and subareas: an example

mobile client side. Moreover, the mobile phone of each participant has a set of sensors, which can complete different sets of tasks (sensor configuration). An appropriate subset of tasks is assigned to each participant. The pre-assigned tasks will only be pushed to the participants online when they connect to the cell towers (e.g., placing a phone call or sending a textual message). After noticing the recommended tasks, the participants complete them with a probability (completion likelihood).

3.2 Sensing Quality and Overall Utility

In order to define the overall system utility, we first need to characterize the sensing quality of multiple concurrent and heterogeneous PS tasks. As different types of tasks require a different number of samples in a subarea, we set a range $(MinT_x, MaxT_x)$ for a specific task x, where $MinT_x$ and $MaxT_x$ are the task-specific minimum and a maximum number of samples for a subarea in a cycle, respectively. Based on the thresholds, we define the following concepts.

Definition 1 (Effective Number of Samples) *Given a specific multi-task allocation plan ψ, the effective number of samples $ES^x_{i,j}$ for a specific task x on a subarea i and in a cycle j is defined as follows:*

$$ES^x_{i,j}(\Psi) = \begin{cases} 0, AS^x_{i,j}(\Psi) < MinT_x \\ min\{AS^x_{i,j}(\Psi), MaxT_x\}, AS^x_{i,j}(\Psi) \geq MinT_x \end{cases} \tag{1}$$

where $AS^x_{i,j}(\Psi)$ is the actual number of samples for a specific task x on a subarea i and in a cycle j.

The intuition behind the above definition of maximum and minimum thresholds is that, on each subarea and in each cycle, a number of measurements are required, which is task-specific. First, at least a number of samples is needed (i.e., the minimum threshold) to guarantee the reliability of collected data. If the minimum requirement cannot be reached, the collected data should not be trusted and is regarded as meaningless, so that the effective number of samples is set to zero. If the minimum requirement is met, the quality would increase as the number of samples increases. However, if it increases beyond a certain value (i.e., the maximum threshold), the data quality might not increase anymore [8], [11]. Note that if the minimum value of a task is set to be 0, then $ES^x_{i,j}(\Psi) = min\{AS^x_{i,j}(\Psi), MaxT_x\}$. Based on the definition of $ES^x_{i,j}(\Psi)$, the sensing quality of a specific task on a spatio-temporal cell is defined as follows.

Definition 2. ($Cell\_DQI$, Data Quality Index per Cell) *Given a task allocation plan Ψ, the sensing quality of a specific task x on a spatio-temporal cell (subarea i and cycle j) is defined as $Cell\_DQI$(Data Quality Index per Cell) as follows:*

$$Cell\_DQI^x_{i,j}(\Psi) = \frac{ES^x_{i,j}(\Psi)}{MaxT_x} \tag{2}$$

Even for the same task, the requirements in dimensions of the time and space are also different. We take the traffic detection task as an example. In terms of space, it requires more data samples for the subareas in downtown than those in the suburb. In terms of time, it requires more samples in rush hours than at night. To this end, we introduce the following concepts to model the spatio-temporal heterogeneity.

Definition 3 (Spatio-temporal Weight) *The importance of each spatio-temporal cell is characterized by the weight factor denoted as $\delta^x_{i,j}$. It is calculated based on the historical query frequency of a PS task and normalized into [0, 1]. First, we calculate the average number data queries in each spatio-temporal cell (subarea i and cycle j) for each task x, which is denoted as $avg^x_{i,j}$. Then, the spatio-temporal weight factor $\delta^x_{i,j}$ is defined as follows:*

$$\delta^x_{i,j} = \frac{avg^x_{i,j}}{\Sigma_{i,j} avg^x_{i,j}} \tag{3}$$

Based on definition 2 and 3, this paper proposes novel metrics to evaluate the data quality for multiple concurrent PS tasks, named Data Quality Index (DQI), which is described as follows:

Definition 4 (DQI, Data Quality Index). *Data quality of a specific task x achieved by a certain task allocation plan Ψ is defined as the weighted sum of the data quality per spatio-temporal cell (i.e., $Cell\_DQI^x_{i,j}$ defined in definition 2, and the weight is the $\delta^x_{i,j}$ described in definition 3). The mathematical formula of DQI of task can be defined as follows:*

$$DQI^x(\Psi) = \Sigma_{i,j} Cell\_DQI(\Psi) * \delta^x_{i,j} \tag{4}$$

In this way, the achieved data quality of a task x ranges from 0 to 1, where 0 indicates that requirements for task x are not satisfied at all, while 1 means that requirements for task x are fully satisfied[2].

Definition 5 (Overall System Utility) *PS system's overall utility achieved by task allocation plan Ψ is defined as the weighted sum of DQI of all tasks, which is formulated as follows:*

$$U(\Psi) = \Sigma_{x=1}^{x=K} DQI^x(\Psi) * w_x \tag{5}$$

where w_x is the weight characterizing the importance of each task and their sum is kept as 1, and K is the total number of tasks. In realistic scenarios, the weights are relevant to many factors. For example, we can simply

[2] For example, in order to calculate $avg^x_{i,j}$, whose corresponding subarea is i and cycle is j (8:00-9:00 am), we will consider the task x's historical query data of subarea i and the same period time (8:00-9:00 am) in the historical data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2869387, IEEE Transactions on Mobile Computing

consider that it is in proportion to the payment of each task, or we can take many others factors (e.g., social benefit) into account. However, this paper does not focus on how to determine the weights. Instead, we assume that it has already been pre-defined by the multi-task PS platform.

3.3 Participant-Side Factors

There are H participants on the platform denoted by $P = \{p_1, \ldots, p_H\}$ and a total of K PS tasks denoted by the $T = \{t_1, \ldots, t_k\}$ We denote all task-and-participant pairs as $V = P \times T$, where $(p_u, t_v) \in V$. In this work, we consider the participant-side factors including sensor configuration, user bandwidth, user availability, and task complete likelihood, which is characterized as follows.

First, based on the sensor configuration of each participant's mobile device, we further select valid task-and-participant pairs and denote it as $V_f = \{(p_u, t_v) \in V | p_u$*'s mobile device is embedded the sensor to complete* $t_v\}$.

Second, the bandwidth of participant p_u is the maximum number of assigned tasks that p_u is willing to accept, which is denoted as L_u.

Third, even with the required sensors, participants may not want to be disturbed by task assignments during some time intervals or at some locations. Thus, PSTasker allows each participant set personalized cycles or subareas where he/she is not willing to receive tasks, which is called the participant availability in this paper. Specifically, we define a spatio-temporal matrix for each participant denoted as $\beta_{u,i,j}$, where $\beta_{u,i,j}$ represents whether a participant p_u allow the system assign tasks to him/her in subarea i and cycle j *(1: allowed, 0: not allowed).*

Finally, in realistic application scenarios, even if in those spatio-temporal cells where the participants allow the task assignments, they will not necessarily complete the assigned tasks. For example, he/she suddenly has something urgent to handle so that fails to get the task done. Thus, given this uncertainty due to human behavior, we use probabilistic modeling to include the participants' task completion likelihood. The recent literature [37], [38] indicate that participants of online crowdsourcing usually complete the assigned tasks with a certain probability. In these works, they assume that participants have a fixed task completion probability (say 0.5). However, PS tasks are different from generic online crowdsourcing tasks in that PS tasks require the participants physically move to certain locations for completing the task. Even if for the same participant, his/her task completion likelihood is diverse among different locations [39]. For example, the empirical study in [39] further demonstrates that participants are more willing to complete tasks in close proximity to their homes. Therefore, instead of setting a fixed probability for each participant as [37], [38], PSTasker characterizes the location-based task completion likelihood. We use $\check{r}_{u,i}$ to represent the task completion likelihood for participant p_u in subarea i. Different from the availabilility constraints, which is participant-defined and deterministic, the task completion likelihood is learned from the participants' historical participation records.

3.4 Problem Definition

A task allocation plan Ψ can be abstracted as a set of task-and-participant pairs, which is called the task allocation instance set (TAIS), which is formally defined in section 4. By taking the above participant-side factors into account, the objective of the research work is to assign an appropriate set of tasks to each participant offline for maximizing the overall system utility while satisfying constraints of maximum bandwidth, availability, and sensor configuration. Meanwhile, it assumes that participants complete assigned tasks with personalized and location-based likelihood. The offline multi-task assignment problem can be formulated as determining a subset of task-participant pairs V_f (denoted as $V_f^* \subseteq V_f$, if $(p_u, t_v) \in V_f^*$, then it means that we assign task t_u to participant p_u) to maximize the overall system utility defined in equation (5), subject to various constraints, which is denoted as follows:

$$Maximize\ \Sigma_{x=1}^{x=K} DQI^x(V_f^*) * w_x \tag{6}$$

$$Subject\ to\ \Sigma_{v=1}^{v=K} y_{u,v} \leq L_u \tag{7}$$

where L_u is the maximum bandwidth of participant p_u, and $y_{u,v}$ indicates whether the task t_v is assigned to the participant u ($y_{u,v} = 1$ if task t_v has been assigned to the participant p_u, and otherwise $y_{u,v} = 0$).

4 Task Allocation Framework

4.1 Framework Overview

PSTasker follows a centralized approach, where a central server collects and stores the volunteering mobile users' historical call/mobility traces in the target area, and the server selects task-and-participant pairs. PSTasker adopts a two-phase procedure to determine the task allocation plan (see Figure 2).

- *Phase 1: Participant profiling.* PSTasker establishes the participants' profiles based on the sensor configuration, mobility traces, user availability settings and users' participation history. This profile is used in estimating the overall utility given a set of task-and-participant pairs (or a TAIS).
- *Phase 2: Task allocation instance set selection.* PSTasker converts the multi-task allocation problem into the representation of a bipartite graph and performs the task allocation in the following two steps. It first selects participants whose bandwidth is not less than the number of tasks the participants can complete according to the sensor configurations (named underloaded participants) and determines their task assignments, which reduces the complexity of the original problem. It then takes an iterative greedy process to optimize the allocation for the rest of participants (named overloaded participants). In each of the iterations, it incrementally selects task-and-participant pairs based on the estimated overall system utility.

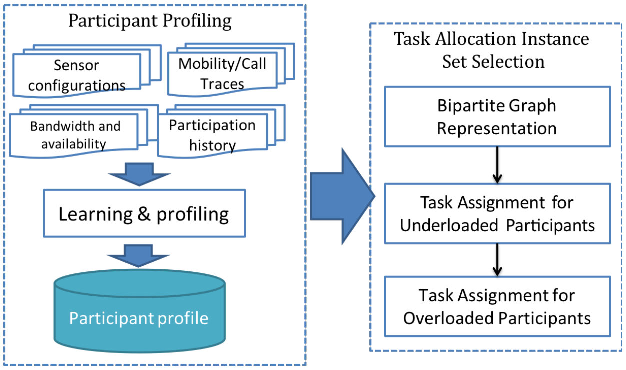

Participant Profiling
Sensor configurations
Mobility/Call Traces
Bandwidth and availability
Participation history
Learning & profiling
Participant profile
Task Allocation Instance Set Selection
Bipartite Graph Representation
Task Assignment for Underloaded Participants
Task Assignment for Overloaded Participants

Fig. 2. PSTasker: framework overview

4.2 Participant Profiling

In this subsection, we describe how the participants' profiles are pre-constructed.

The participant profiling can be divided into the following two categories. One is directly captured from participants' mobile device or their self-defined information, and the other is learned from historical task participation and mobility records.

(1) Directly-Captured Profile

First, PSTasker allows each participant pre-define their bandwidth and availability. The bandwidth refers to the self-defined maximum number of tasks each participant is willing to complete. Additionally, participants may not be willing to receive tasks during some periods of time or at some locations, so that we allow each participant to define their blocked spatio-temporal cells (i.e., availability).

Second, the participant's mobile device might not possess the required sensors, or some sensors may be disabled by the participants. To reduce the participants' burden, PSTasker automatically captures sensor configurations when the participants registered their mobile devices on the PS platform.

(2) Learning-based Profile

First, PSTasker learns the task completion likelihood $\check{r}_{u,i}$ based on the historical participation records of the participant p_u. We assume that, according to the historical participation records, the number of tasks participant p_u receives in subarea i is $q_{u,i}$, while the number of completed tasks is $COM_{u,i}$, so that we calculate the location-based task completion likelihood of participant p_u in subarea i as $\check{r}_{u,i} = COM_{u,i}/q_{u,i}$.We also notice that some participants may not have historical participation records in certain locations. In this case, we calculate it based on his/her overall participation records in all locations, i.e., $\check{r}_{u,i} = \Sigma_i COM_{u,i}/\Sigma_i q_{u,i}$.

Second, we predict the number of samples during cycle j at subarea i by calculating the probability of each participant connecting at least one time to the tower in each cycle. We first map each participant's historical call traces onto N sensing cycles. Then it counts the average number of connection by each participant p_u at each subarear i in each cycle j, which is denoted as $\lambda_{u,i,j}$ For example, to estimate $\lambda_{u,i,j}$ for cycle j from 08:00 to 09:00, we will count the average number of connections by p_u at i during the same period 08:00-09:00 in the historical records. Assuming the connection sequence follows an inhomogeneous Poisson process [11], the probability of a participant p_u to connect n times to cell tower i in cycle j can be modeled as:

$$Pro_{i,j}(u,n) = \frac{\lambda^n_{u,i,j}}{n!} \times e^{-\lambda_{u,i,j}} \tag{8}$$

Therefore, we can estimate the probability of participant p_u connecting at least one time during cycle j at cell tower i as follows:

$$Pro_{i,j}(u) = \Sigma_{n=1}^{\infty} Pro_{i,j}(u,n) = 1 - e^{-\lambda_{u,i,j}} \tag{9}$$

Thus, we predict that the probability of participant p_u providing one sample for each assigned task during cycle j at cell tower i as:

$$\alpha_{i,j}(u) = Pro_{i,j}(u) = 1 - e^{-\lambda_{u,i,j}} \tag{10}$$

4.3 Utility Estimation

The objective is to find a set of task-and-participant pairs offline that can maximize the overall utility online. Therefore, we need to estimate the overall utility of different combinations in one unified function. By further considering different participant-side factors (including task completion likelihood, availability constraints, and mobility patterns), given a set of task-and-participant pairs (or a TAIS) denoted as $V' \subseteq P \times T$, the estimated actual number of samples for task x in subarea i and cycle j can be estimated as follows.

$$AS^x_{i,j}(V') = \Sigma_{(p_u,x)\in V'}\{(1 - e^{-\lambda_{u,i,j}}) * \check{r}_{u,i} * \beta_{u,i,j}\} \tag{11}$$

According to definition 1 and equation (1), we can get the effective number of sumaple $ES^x_{i,j}(V')$ by comparing $AS^x_{i,j}(V')$ with $MaxT_x$ and $MinT_x$.

Finally, according to definition 2, 4 and 5, the overall utility achieved by TAIS V′ can be calculated as:

$$\Sigma_{x=1}^{x=K}(\Sigma_{i,j}\frac{ES^x_{i,j}(V')}{MaxT_x} * \delta^x_{i,j}) * w_x \tag{12}$$

4.4 Task Allocation Instance Selection

(1) Bipartite Graph Representation PSTasker first converts the original multi-task assignment problem into a representation of a bipartite graph. Specifically, given a participant set $P = \{p_1, \ldots, p_H\}$ and a PS task set $T = \{t_1, \ldots, t_K\}$, we denote each participant/task as a vertex in the bipartite graph, where participant and task vertices have distinct vertex types. There exists an edge between a participant vertex p_u and a task vertex t_v, if and only if participant p_u can complete task t_v according to the sensor configuration profile. We say that the task-and-participant pair (p_u, t_v) is valid, if there is an edge between vertices p_u and t_v in the graph. Based on the definition of valid pairs, we define the concept of task allocation instance set as follows. The bipartite graph consisting of all valid task-and-participant pairs is called the original graph, which is denoted as G_0.

Definition 6: Task allocation instance set. Given an original graph G_0 consisting of participant set P and a task set T, a task allocation instance set (TAIS), denoted by I, is a set of task-and-participant pairs in the form (p_u, t_v),

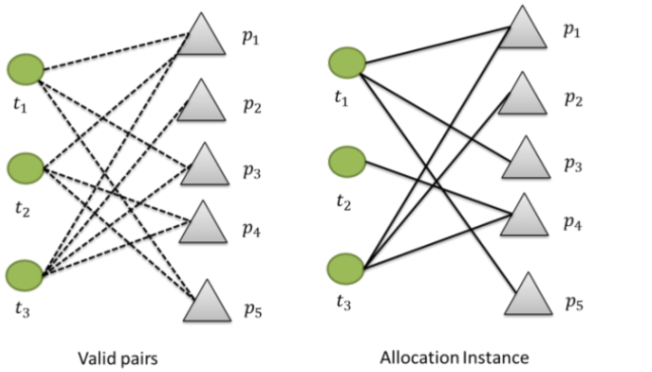


Fig. 3. Illustration of the task-and-participant assignment: an example

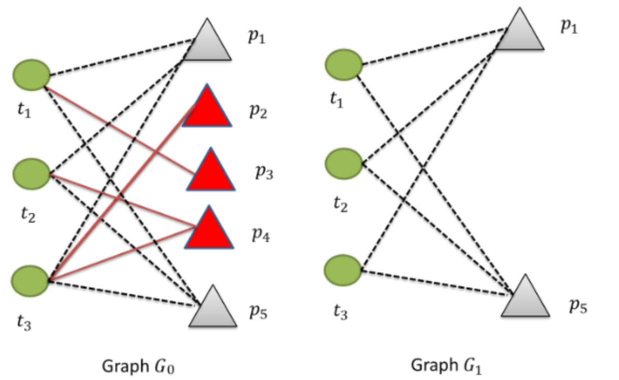


Fig. 4. Task assignment for underloaded participants: an example

where each (p_u, t_v) is a valid pair and the degree of vertex p_i (denoted as $deg(p_u)$) subject to $deg(p_u) \leq L_u$, where L_u is the pre-defined maximum bandwidth of participant p_u.

Figure 3 (left) illustrates an example of valid pairs (i.e., original graph G_0), where five available participants and three PS tasks are denoted by triangular and circular nodes, respectively, and valid task-and-participant pairs are represented by dashed lines. If we assume that the maximum bandwidth of each participant is two, then the **Figure 3** (right) depicts the result of one possible task allocation instance, where the bold lines indicate assignment pairs. Then, the problem formulated above can be converted into determining a task allocation instance set in an original bipartite graph G_0 consisting of multiple valid task-and-participant pairs to maximize the overall system utility.

(2) Task Assignment for Underloaded Participants In this step, we first divide the participant vertices into two categories (i.e., the overloaded participant vertexes and underloaded participant vertexes) based on if a participant is competed by different tasks. We search the original graph and count the degree of each participant vertex p_u (denoted as $deg(p_u)$)). Based on the relationship between $deg(p_u)$) and the maximum bandwidth of p_u , we define these two types of vertexes as follows.

Definition 7 (Two Types of Participant Vertexes). Participant vertexes subject to $deg(p_u) > L_u$ (i.e., vertexes whose degree is more than its maximum bandwidth) is called the *overloaded participant vertexes.* Otherwise, the participants are referred to as the *underloaded participant vertexes.*

Intuitively, the underloaded participants are those whose maximum bandwidth is not less than the number of tasks he/she is capable of completing according to sensor configuration, while the others are the overloaded ones. For example, if a participant's profile (sensor configuration and spatio-temporal availability) indicates that he/she can only complete two of the tasks in the system and he/she sets the maximum bandwidth as three, then the participant is underloaded.

After identifying underloaded vertexes, we select all their connected edges in into the TAIS, which means PSTasker just assigns all feasible tasks to him/her. After the tasks are assigned, we delete these underloaded vertexes and assigned edges, because we have already taken full use of their sensing resources. After the above deletion, if the degree of a task vertex drops to 0, then it is deleted because none of the remaining participants can complete it.

Figure 4 (left) shows an example of an original bipartite graph consisting of 3 tasks and 5 participants. If we assume that bandwidth of each participant is set as 2, then the red triangular nodes (p_2, p_3 and p_4) are the underloaded vertices and the rest are overloaded vertexes (p_1 and p_5). After the task assignment and deletion of red vertexes and red edges, the rest overloaded vertexes and corresponding edges form a new graph denoted as G_1 (see **Figure 4** (right)). We can see that the original problem is simplified after this step.

(3) Task Assignment for Overloaded Participants The goal of this step is to select edges in the new graph (e.g., the graph G_1 in **Figure 4**), and combine it with selected edges to maximize the expected overall system utility.

We first consider a special case, where the minimum threshold $MinT_x = 0$ for each task. In this case, the effective number of samples in each spatio-temporal cell defined in Equation (1) is re-defined as $ES^x_{i,j} = min\{AS^x_{i,j}, MaxT_x\}$. We can adopt a naive greedy algorithm, which incrementally selects edges based on the estimated overall system utility. Initially, we set the TAIS V' as the edges selected in the step of task assignment for the underloaded participants. Then, we would incrementally select one edge with the highest utility increase and add it to set V'. Given an edge (p_u, t_v), its utility increase can be calculated according to equation (13). The greedy selection process repeats until the bandwidths of all participants have been used up, or the participants with remaining bandwidth are not willing to complete any tasks according to their availability constraints.

$$\begin{aligned}&\Sigma_{x=1}^{x=K}(\Sigma_{i,j}\frac{ES^x_{i,j}(V'\cup(p_u,t_v))}{MaxT_x}*\delta^x_{i,j})*w_x-\\&\Sigma_{x=1}^{x=K}(\Sigma_{i,j}\frac{ES^x_{i,j}(V')}{MaxT_x}*\delta^x_{i,j})*w_x\\&=\Sigma_{i,j}\{Cell\_DQI^v_{i,j}(V'\cup(p_u,t_v))-Cell\_DQI^v_{i,j}(V')\}\end{aligned} \tag{13}$$

However, in general cases, where $MinT_x \neq 0$, it is likely that none of the task-and-participant pairs leads to an overall utility increase when adding to the set during earlier rounds of the greedy selection process, as $MinT_x$ of each task has not been reached. One direct adjustment is to randomly select task-and-participant pairs in earlier iterations. With the additions of more task-and-participant

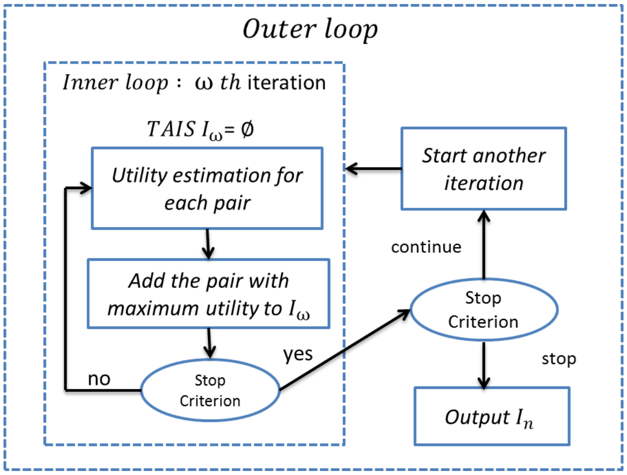


Fig. 5. Nested-loop procedure for task assignment of overloaded participants

pairs, tasks with lower $MinT_x$ would reach the thresholds earlier, so that the algorithm tends to allocate more limited resources (i.e., the participants) to those tasks. Nevertheless, this may have a negative impact on the overall utility optimization. For example, when tasks with lower $MinT_x$ 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 participants, which is bad for the overall utility maximization. Therefore, when $MinT_x \neq 0$, we need to improve the above naive greedy method by introducing more sophisticated strategy.

Thus, to reduce the impact of $MinT_x$ on resource allocation, we propose an iterative greedy procedure in **Figure 5**. The procedure is nested-loop, which consists of inner and outer loops.

In the first iteration of the outer loop, PSTasker uses the naive greedy algorithm to get the initial task allocation instance set denoted as I_1, in which it iteratively selects the edges with highest overall utility increase only based on Equation (13). Then, starting from the second iteration of the outer loop, we also adopt the similar greedy process but estimating the utility increase differently for a task-and-participant pair. Specifically, we first try to estimate its utility increase according to Equation (13). If the increase is not zero, then it is regarded as the estimated utility increase of this pair. Otherwise, we give another chance to estimate it based on the obtained TAIS in the previous iteration. The stopping criterion of the outer-loop iterations is that the estimated system utility at the end of current iteration does not increase compared to the previous iteration. For the inner loop, the stopping criterion is that the bandwidths of all participants have been used up, or the participants still with remaining bandwidth cannot complete any tasks according to their pre-defined availability constraints. The pseudocode of the above process is illustrated as follows.

Finally, we analyze the time complexity of the above iterative greedy process as follows. The time cost of computing valid task-and-participant assignment pairs is given by $O(K * H)$ in the worst case, where any of H participants can be assigned to any of K tasks (i.e., $K * H$ valid task-and-participant pairs). Moreover, since each participant is assigned to at most L_i tasks, the number of rounds in each of the iterations is at most ΣL_i times. Therefore, the total time complexity of each of the outer-loop iterations can be given by $O(K * H * \Sigma L_i)$. Assume that the total number of outer loop iterations is ω, then the total time complexity of the iterative greedy process's is $O(\omega * K * H * \Sigma L_i)$. According to our extensive experiment, the outer loop typically runs iterations 5∼10 iterations, so that it is polynomial time algorithm.

Algorithm 1 task assignment for overloaded participants

```
Input: $I_1, P, V', L$
Output: $I_\omega$
 1: $\omega = 1$
 2: repeat
 3:     for each $(p_u, t_v)$ do
 4:         $\Delta_{u,v} = \Sigma_{i,j}\{Cell\_DQI^v_{i,j}(V' \cup (p_u, t_v)) - Cell\_DQI^v_{i,j}(V')\}$
 5:         $u, v = argmax\Delta_{u,v}, I_\omega = I_\omega \cup (p_u, t_v), L_u = L_u - 1$
 6:         if $L_u = 0$ then $P = P - \{u\}$
 7:         end if
 8:     end for
 9: until $P = \varnothing$
10: $\omega = 2$
11: repeat
12:     for each $(p_u, t_v)$ do
13:         $\Delta_{u,v} = \Sigma_{i,j}\{Cell\_DQI^v_{i,j}(V' \cup (p_u, t_v)) - Cell\_DQI^v_{i,j}(V')\}$
14:         if $\Delta_{u,v} = 0$ then
15:             if $(p_u, t_v) \cap I_{\omega-1} = \varnothing$ then
16:                 $\Delta_{u,v} = \Sigma_{i,j}\{Cell\_DQI^v_{i,j}(I_{\omega-1} \cup (p_u, t_v)) - Cell\_DQI^v_{i,j}(I_{\omega-1})\}$
17:             else
18:                 $\Delta_{u,v} = \Sigma_{i,j}\{Cell\_DQI^v_{i,j}(I_{\omega-1}) - Cell\_DQI^v_{i,j}(I_{\omega-1} - \{(p_u, t_v)\})\}$
19:             end if
20:         end if
21:         $u, v = argmax\Delta_{u,v}, I_\omega = I_\omega \cup (p_u, t_v), L_u = L_u - 1$
22:         if $L_u = 0$ then $P = P - \{u\}$
23:         end if
24:     end for
25: until $P = \varnothing$
26: if the overall utility is increased then
27:     $\omega = \omega + 1$, go to line 11
28: end if
29: return $I_\omega$
```

5 Performance Evaluation

5.1 Datasets and Experimental Settings

In this paper, we evaluate the performance of PSTasker based on D4D, an open and real-world mobility trace dataset [36]. The original D4D datasets involve 50000 users' phone records of connections to the cell towers (e.g. making phone calls or sending text messages) from Cote d'Ivoire, in which each record includes user id, connection time and cell tower. We extract and generate the datasets we need based on D4D datasets in the following steps.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2869387, IEEE Transactions on Mobile Computing

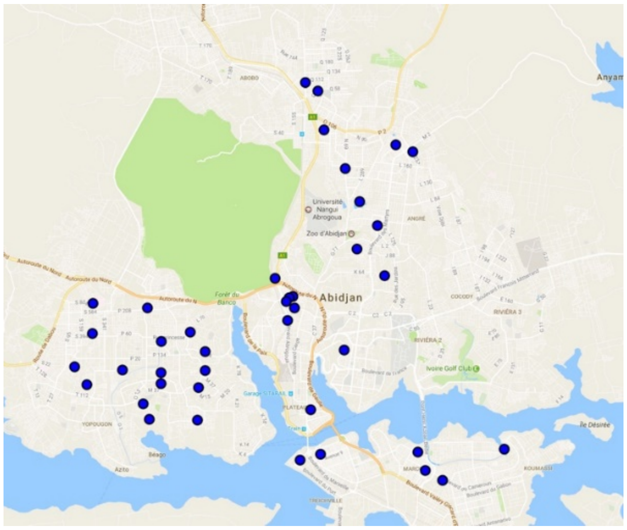


Fig. 6. Cell towers used in our experiment

TABLE 2
Parameter settings

Parameter	Settings
$MinT_x, MaxT_x$	$MinT_x : (0, 3), MaxT_x : (MinT_x, 5)$
K	Number of tasks $K = 10, 20, 50$
H	Number of participants $H = 1000, 1500, 2000, 2500$
$\sigma^x_{i,j}$	Randomly generated from (0,1) while keeping the sum of the different spatio-temporal cell as 1.
L_u	Randomly generated based on the Gaussian distribution with mean value denoted as μ
μ	$\mu = 4, 6, 8, 10$
$\check{r}_{u,i}$	$base(u) * (1 + LocVariance(i))$
$U(\Psi)$	PS system's overall utility achieved by task allocation plan Ψ
τ	Randomly generate from (0.1, 0.5)
$base(u)$	Randomly generated based on Gaussian distribution with mean value B_{mean}
B_{mean}	$B_{mean} = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$

We first re-select the users randomly every 2 weeks and totally 10 two-week periods of records are prepared. In each two-week dataset, we use the mobility traces in the first week to execute our task allocation algorithm and obtained a TAIS. Then, we test the overall system utility of this TAIS using the second-week dataset. Specifically, we extracted records of the downtown area with 40 cell towers (Figure 6), where the mobile users are densely distributed. We further assume that each task executes for 5 days (weekdays in a week), lasts for 10 sensing cycles every working day from 8:00 am to 6:00 pm, with each cycle lasting one hour. Thus, the total period consists of 50 sensing cycles.

Then, we simulate other parameters for both PS tasks and participants. In terms of tasks, we randomly generate the weights of each task and each spatio-temporal cell, minimum and maximum threshold value (minimum: from 0 to 3, maximum: from minimum to 5). In terms of participants, we simulate different aspects of their profiles as follows. First, we randomly generate $\beta_{u,i,j}$ and feasibility for all task-participant pairs (feasible/infeasible) as 0 or 1. Second, we assume that the participant's sensing bandwidth L_u follows the Gaussian distribution with mean value μ. Third, as participants are more willing to complete tasks in close proximity to their homes than other places [39], we generate the location-based task completion likelihood as $\check{r}_{u,i} = min\{base(u) * (1 + LocVariance(u, i), 1)\}$, where $base(u)$ of each participant follows Gaussian distribution with mean value B_{mean}, and $LocVariance(u, i) = \tau$ if subarea i is where the participant's residence place is located, otherwise, $LocVariance(u, i) = 0$ Though the experimental result in [39] indicates that people are more likely to complete tasks at or close to home, we cannot find literature which gives a specific value for τ. Thus, we generate τ from (0.1, 0.5) randomly. For example, if $base(u)$ of participant p_u is generated as 0.7 and τ is generated as 0.15, p_u will complete tasks with the probability of 0.7*1.15 in subareas where the participant's home is located and with the probability of 0.7 in other subareas.

According to [40], home locations are the most frequently visited stop during nighttime, so we simulate the location of participants' residence place based on the following rule: if the participants make most of the phone calls between 10:00pm 4:00 am in a subarea i, then the subarea i is regarded as the participants' residence place. Before executing the task allocation algorithm, we assume that the generated $\check{r}_{u,i}$ has already been learned from the historical participation records.

The multi-task allocation in this paper can be seen as a resource allocation problem, where the demand and supply of the resource may affect the system's performance. Thus, based on the above original and generated datasets and experimental settings, we evaluate the PSTasker's performance by varying the number of participants/tasks, maximum bandwidth, and task completion likelihood. Table 2 summarizes the above parameter settings.

5.2 Baselines

We then provide the following baseline methods for comparative studies.

- *Random Allocation (RA)* - This method randomly assigns tasks to each participant based on all participant-side constraints. As the order of participants may have an impact on the performance, we repeat the random allocation process for 30 times, each of which is based on a randomly generated participants' order.
- *Naive Greedy Allocation (NGA)* - This method adopts a simple greedy algorithm to perform task allocation. It incrementally selects task-and-participant pairs based on the estimated overall system utility increase using Equation (13). If the utility increase of all task-and-participant pairs is zero, we randomly select one and add into the TAIS.
- *Naive Greedy by Concave-relaxation (NGCR)* - Similar to NGA, this method also incrementally selects task-and-participant pairs with highest utility increase. However, when estimating the utility increase of a given pair, it adopts a different utility function with a concave-relaxation, which changes the Equation (1) into $ES'^{x}_{i,j} = min\{AS^x_{i,j}, MaxT_x\}$.

5.3 Performance Comparison

In this subsection, we evaluate the performance of PSTasker and other baseline methods by varying key parameters. We carried out experiments using a laptop with an Intel Core i7-4710HQ Quad-Core CPU and 16GB memory. PSTasker and baseline algorithms were implemented with the Java SE platform on a Java HotSpotTM 64-Bit Server.

(1) Different number of tasks

Figure 7a and Figure 8a present the performance comparison on the overall utility and running time among PSTasker and other baseline methods under a different number of tasks, where we fixed the number of participants to 2000, mean value μ to 8 and B_{mean} to 0.7.

From Figure 7a, we can see that PSTasker outperforms RA, NGA and NGCR by achieving 7.3%∼15.1% higher overall utility for a various number of tasks, respectively, and the advantage becomes more significant as the number of tasks increases. 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, PSTasker can better support such practical scenarios.

Figure 8a shows the running time of different methods. The fastest algorithm is RA. The running time of PSTasker is longer than NGA and NGCR. Although PSTasker needs longer running time than other baselines, its running time is less than 10 minutes for a various number of tasks. Considering that the algorithm is executed offline, the computation time is reasonable. Furthermore, if we deploy PSTasker onto a real-world PS system, then shorter computation time can be obtained by using parallel algorithms or using a powerful commercial server.

(2) Different number of participants

Figure 7b and Figure 8b presents the performance comparison on the overall utility and running time among different methods under a different number of participants, where we fix the number of tasks to 20, mean value μ to 8 and value of B_{mean} to 0.7.

Figure 7b indicates that the overall utility increases with the number of participants for all methods. This is because, with increasing number of participants, the shared resources become more abundant. PSTasker outperforms RA, NGA and NGCR by achieving 10.9%∼25% higher overall utility for a various number of participants.

Figure 8b reports the running time of different algorithms, and we can see that the running time PSTasker needs is longer than other baselines methods. However, the running time of PSTasker is less than 10 minutes for a various number of participants, which is acceptable as it is executed offline.

(3) Different values of μ

When varying the parameter μ, we fix the number of tasks to 20, the number of participants to 2000 and value of B_{mean} to 0.7. Figure 7c shows the overall utility for different values of μ. From the figure, we can see that under various settings of mean value μ, PSTasker outperforms RA, NGA and NGCR by achieving 9.7%∼18.3% higher overall utility. Figure 8c shows the running time of PSTasker is longer than other baseline methods but less than 10 minutes.

(4) Different values of B

When varying B_{mean}, we fix the number of tasks as 20, the number of participants to 2000 and bandwidth's mean value μ to 8. From Figure 7d, we can see that, with higher task completion likelihood B_{mean}, the overall utility of different approaches is improved in general. This is because the higher the task completion likelihood is, the more predictable the participants become. The PSTasker consistently outperform the other baselines by achieving 8.0%∼14.1% higher overall utility when B_{mean} varies. In addition, Figure 8d shows that the completion likelihood does not have an obvious impact on the running time of different approaches.

5.4 Detailed Analysis

The above experimental results demonstrate that PSTasker outperforms other baseline methods under various settings. In this subsection, we will give more details to clarify the reason why PSTasker can outperform them.

First, we investigate why PSTasker outperforms NGA. We demonstrate the number of assigned task-and-participant pairs for tasks with different $MinT_x$. Due to the space limit, we only show the result under a specific setting (20 tasks and 2000 participants) in Figure 9.

From the figure, we can see that the tasks with lower $MinT_x$ obtain more assigned participants. In particular, this trend is more obvious for NGA than PSTasker. By reviewing the optimizing process of NGA, we find that for tasks with lower $MinT_x$, with the adding of task-and-participant pairs, the number of estimated sensor readings in spatio-temporal cells would reach to $MinT_x$ earlier. Therefore, the participants are more likely to be assigned to these tasks, while other tasks have fewer assigned ones. From the definition of overall system utility, we can see that if the tasks with lower $MinT_x$ are less important (i.e., with lower weight), assigning most of the limited participants to these tasks counts against the overall utility optimization. On the contrary, PSTasker adopts an iterative process, where the estimation of utility increase is based on the solution in the previous iteration, so that tasks with higher $MinT_x$ gain another opportunity get participants assigned. In summary, PSTasker outperforms the NGA in that its iterative greedy process reduce the impact of $MinT_x$ on resource allocation.

Next, we discuss why PSTasker outperforms NGCR. We analyze the actual number of samples $AS^x_{i,j}$ in each spatio-temporal cell for different tasks. Specifically, the spatio-temporal cells are divided into three categories: Category-1: $AS^x_{i,j} = 0$; Category-2: $0 < AS^x_{i,j} < MinT_x$; Category-3: $AS^x_{i,j} \geq MinT_x$ The datasets we used in the experiment are related to 40 subareas and 50 cycles so that there are 2000 total spatio-temporal cells. Figure 10 shows the distribution of the above three classes (20 tasks, 2000 participants, B_{mean} and μ=8).

Intuitively, the spatio-temporal cells with Category-2 have a negative effect on the optimization result because it indicates wasted resources that consumed but do not help to increase overall utility. From the figure, we can see that the number of cells with Category-2 obtained by PSTasker is significantly lower than NGCR, while the number of cells with Category -3 obtained by PSTasker is higher than NGCR. Therefore, the advantage of PSTasker (compared to NGCR) lies in the fact that it reduces resource waste.

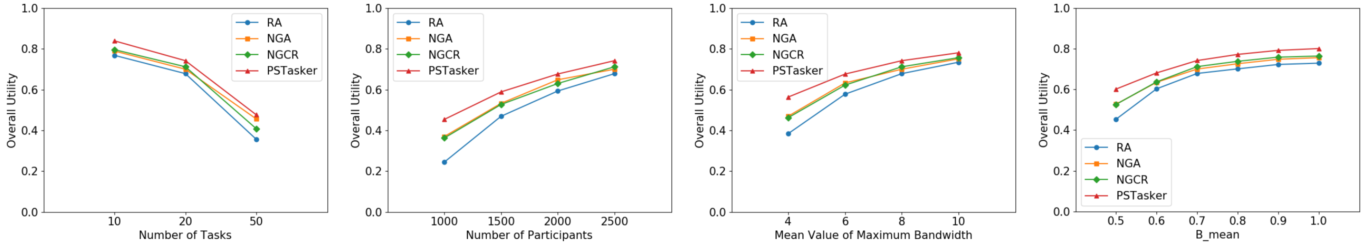


Fig. 7. Overall utility comparison: (a) with various number of tasks (b) with various number of participants (c) with various maximum bandwidth (d) with various B_{mean}

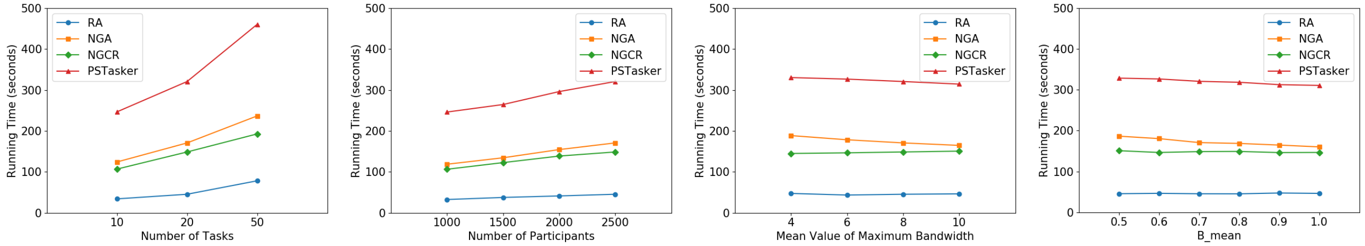


Fig. 8. Running time comparison: (a) with various number of tasks (b) with various number of participants (c) with various maximum bandwidth (d) with various B_{mean}

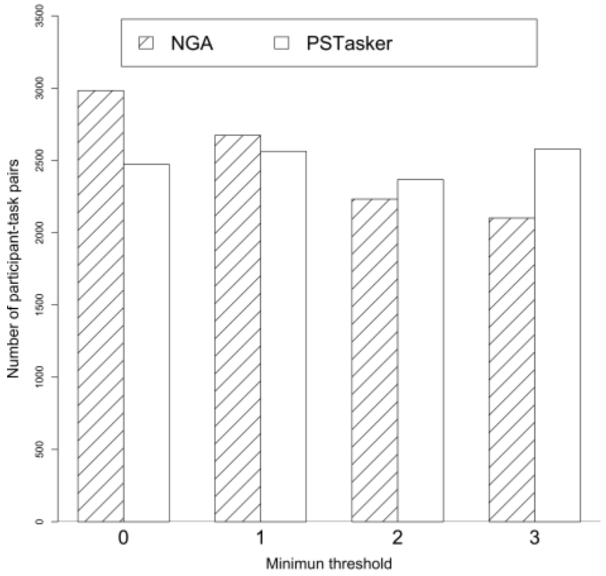


Fig. 9. Number of assigned task-and-participant pairs for tasks with different minimum thresholds

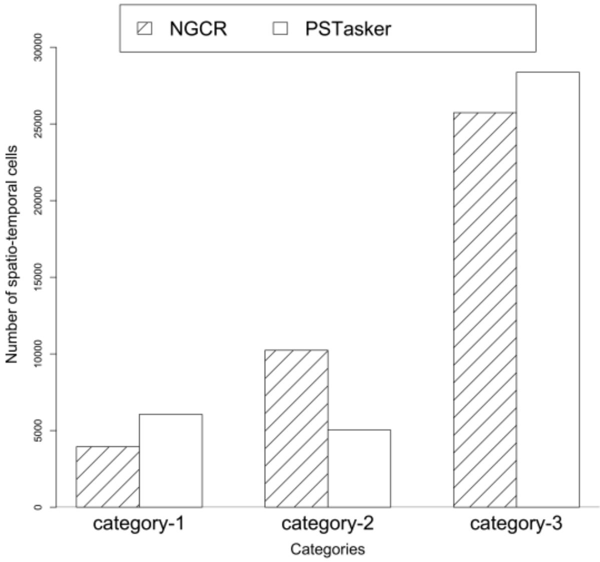


Fig. 10. Sampling status in spatio-temporal cells: NGCR vs PSTasker

6 Limitation and Discussion

We have shown that PSTasker is effective in real-world datasets and simulated parameters, but several issues and limitations remain, which need to be further discussed and explored.

First, the good performance of PSTasker relies on the assumption that the likelihood can be learned accurately. Thus, the first research question we should address in future work is how to model the likelihood with its correlated factors. From state-of-the-art literatures such as [41], [42], we know that many related factors could affect the participants' decision on whether completing a certain task, including both task attributes (e.g., task price, task type, sensing time, consumed energy, etc.) and participant attributes (e.g., demographic, busyness, mental fatigue, etc.). A systematic review is needed before building a more comprehensive model.

Second, even if we are able to build a good model of task completion likehood, it is still very hard to learn it in real-world application scenarios, which will be our future work direction. The first challenge is how to get the value of correlated features. For example, the authors in [42] found that busyness and mental fatigue have significant impact on the task completion likehood. However, how to obtain a participants' busyness and mental fatigue status is very challenging. Sensors in smartphone may provide some opportunity, but it is not a easy task. Aother challenge is that, for new participants without an adequate number of historical participation records, we cannot directly learn their task completion likelihood. In these cold-start cases, we may utilize the records of other participants with similar profiles, and some semi-supervised learning approaches may also be effective.

Third, although we used a real-world mobility dataset in the evaluation section, some relevant parameters about tasks and participants are simulated with certain assumptions (e.g., B_{mean}, μ, $MinT_x$, $MaxT_x$, $LocVariance(i)$). For future work, we are now cooperating with a local government to develop a city-scale PS platform for ordinary citizens to report information (e.g., traffic jams, missing manhole cover, etc.). We intend to test the core algorithm of PSTasker in the platform to identify more practical issues for further improvement.

Fourth, PSTasker considers various factors (e.g., task sensing quality, sensor configuration, spatial-temporal availability, completion likelihood, etc) when formulating the problem and designing the task allocation framework. However, some other factors such as the budget constraints of PS platforms and the interests of participants should also be considered. Furthermore, 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 factors or constraints in our future work.

7 Conclusion

This paper proposes a novel task allocation framework, named PSTasker, for participatory sensing (PS). PSTasker coordinates the allocation of multiple tasks to maximize the overall system utility of the PS platform while considering various participant-side factors, PSTasker adopted an iterative greedy process to optimize the task allocation. Extensive evaluations based on real-world mobility traces demonstrate that PSTasker outperforms the baseline methods under various settings.

References

[1] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, Participatory sensing, in ACM SenSys'06, 2006, pp. 15.

[2] Kanhere, S. S. (2013, February). Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. InInternational Conference on Distributed Computing and Internet Technology(pp. 19-26). Springer Berlin Heidelberg

[3] Campaignr: http://research.cens.ucla.edu/urban/

[4] M. Ra, B. Liu, T. F. L. Porta, and R. Govindan. 2012. Medusa: A programming framework for crowd-sensing applications. In Proceedings of the 10th international conference on Mobile systems, applications, and services (MobiSys '12), 337-350.

[5] Sasank Reddy, Katie Shilton, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using context annotated mobility profiles to recruit data collectors in participatory sensing. In Location and Context Awareness, pages 5269. Springer, 2009.

[6] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In Proceedings of Pervasive, pages 138155. 2010.

[7] Giuseppe Cardone, Luca Foschini, Paolo Bellavista, Antonio Corradi, Cristian Borcea, Manoop Talasila, and Reza Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. Communications Magazine, IEEE, 51(6), 2013.

[8] D. Zhang, H. Xiong, L. Wang, and G. Chen, Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint, in The 2014 ACM Conference on Ubiquitous Computing, UbiComp '14, Seattle, WA, USA, 2014, pp. 703714.

[9] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi, EMC3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint, IEEE Transactions on Mobile Computing, 2015.

[10] H. Xiong, D. Zhang, L. Wang, J. Gibson, and J. Zhu, EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants, ACM Transactions on Intelligent Systems and Technology (TIST), 2015.

[11] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint, in IEEE International Conference on Pervasive Computing and Communications (Percom'15), 2015.

[12] Zhang, M., Yang, P., Tian, C., & Tang, S. (2015). Quality-aware sensing coverage in budget constrained mobile crowdsensing networks.IEEE Transactions on Vehicular Technology, 1-1.

[13] Adish Singla and Andreas Krause. Incentives for privacy tradeoff in community sensing. In First AAAI Conference on Human Computation and Crowdsourcing, 2013.

[14] Song, Z., Zhang, B., Liu, C. H., Vasilakos, A. V., Ma, J., & Wang, W. (2014, June). QoI-aware energy-efficient participant selection. InSensing, Communication, and Networking (SECON), 2014 Eleventh Annual IEEE International Conference on(pp. 248-256). IEEE.

[15] Karaliopoulos, M., Telelis, O., & Koutsopoulos, I. (2015). User Recruitment for Mobile Crowdsensing over Opportunistic Networks. InINFOCOM 2015 Proceedings, IEEE.

[16] S. Hachem, A. Pathak, and V. Issarny. Probabilistic registration for large-scale mobile participatory sensing. In Proceedings of the 2013 IEEE International conference on Pervasive Computing and Communications, volume 18, page 22, 2013.

[17] D. Philipp, J. Stachowiak, P. Alt, F. Durr, and K. Rothermel. Drops: Model-driven optimization for public sensing systems. In proceedings of the 2013 EEE International Conference on Pervasive Computing and Communications, volume 18, page 22, 2013.

[18] Wang, L., Zhang, D., Pathak, A., Chen, C., Xiong, H., Yang, D., & Wang, Y. (2015, September). CCS-TA: quality-guaranteed online task allocation in compressive crowdsensing. InProceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing(pp. 683-694). ACM.

[19] Leye Wang, Daqing Zhang, Yasha Wang, Chao Chen, Xiao Han and Abdallah Mhamed. Sparse Mobile Crowdsensing: Challenges and Opportunities. IEEE Communications Magazine, 2016, vol. 54, no. 7, 2016, pp. 161-167.

[20] Liu, Y., Guo, B., Wang, Y., Wu, W., Yu, Z., & Zhang, D. TaskMe: multi-task allocation in mobile crowd sensing.ACM International Joint Conference on Pervasive and Ubiquitous Computing (Ubicomp 2016).

[21] Guo, B., Liu, Y., Wu, W., Yu, Z., & Han, Q. (2016). ActiveCrowd: A Framework for Optimized Multitask Allocation in Mobile Crowdsensing Systems.IEEE Transactions on Human-Machine Systems.

[22] Wang, Hanshang Li Ting Li Yu. "Dynamic Participant Recruitment of Mobile Crowd Sensing for Heterogeneous Sensing Tasks." 12th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2015).

[23] Wang, J., Wang, Y., Zhang, D., Xiong, H., Wang, L., & Sumi, H., et al. (2016). Fine-grained multi-task allocation for participatory sensing with a shared budget. Internet of Things Journal (in press).

[24] Song, Z., Liu, C. H., Wu, J., Ma, J., & Wang, W. (2014). QoI-Aware Multi-task-Oriented Dynamic Participant Selection With Budget Constraints. Vehicular Technology, IEEE Transactions on,63(9), 4618-4632.

[25] M. Buhrmester, T. Kwang, and S. D. Gosling. 2011. Amazon's mechanical turk a new source of inexpensive, yet high-quality, data? Perspectives on Psychological Science, 6: 3-5.

[26] CrowdFlower: https://www.crowdflower.com/

[27] Samasource: http://www.samasource.org/

[28] Roy, S. B., Lykourentzou, I., Thirumuruganathan, S., Amer-Yahia, S., & Das, G. (2015). Task assignment optimization in knowledge-intensive crowdsourcing.The VLDB Journal,24(4), 467-491.

[29] Assadi, S., Hsu, J., & Jabbari, S. Online assignment of heterogeneous tasks in crowdsourcing markets. HCOMP 2015.

[30] Goel, Nikzad, and Singla, Mechanism Design for Crowdsourcing Markets with Heterogeneous Tasks, HCOMP 2014.

[31] X. Lu, D. Li, B. Xu, W. Chen, and Z. Ding, Minimum cost collaborative sensing network with mobile phones, in IEEE ICC'13, 2013, pp. 18161820.

[32] ang, F., Lu, J. L., Zhu, Y., Peng, J., Shu, W., & Wu, M. Y. Heterogeneous Task Allocation in Participatory Sensing. IEEE GlobeCom 2015.

[33] Jiangtao Wang, Yasha Wang, Daqing Zhang, Feng Wang, Yuanduo He, Liantao Ma: PSAllocator: Multi-Task Allocation for Participatory Sensing with Sensing Capability Constraints. The 20th ACM Conference on Computer-Supported Cooperative Work and Social Computing(CSCW 2017); 02/2017, Portland, USA.

[34] Assadi, S., Hsu, J., & Jabbari, S. Online assignment of heterogeneous tasks in crowdsourcing markets. HCOMP 2015.

[35] Goel, Nikzad, and Singla, Mechanism Design for Crowdsourcing Markets with Heterogeneous Tasks, HCOMP'14.

[36] V.D. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki. Data for development: the d4d challenge on mobile phone data. 2012.

[37] Ikeda, K., & Bernstein, M. S. (2016). Pay It Backward: Per-Task Payments on Crowdsourcing Platforms Reduce Productivity.CHI Conference on Human Factors in Computing Systems. ACM.

[38] Roy, S. B., Lykourentzou, I., Thirumuruganathan, S., Amer-Yahia, S., & Das, G. (2015). Task assignment optimization in knowledge-intensive crowdsourcing.The VLDB Journal,24(4), 467-491.

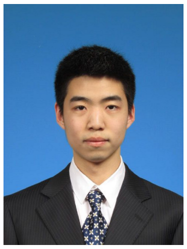
[39] Alt, F., Shirazi, A. S., Schmidt, A., Kramer, U., & Nawaz, Z. (2010, October). Location-based crowdsourcing: extending crowdsourcing to the real world. InProceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries(pp. 13-22). ACM.

[40] Chen, C., Bian, L., & Ma, J. (2014). From traces to trajectories: How well can we guess activity locations from mobile phone traces?. Transportation Research Part C: Emerging Technologies, 46, 326-337.

[41] Thebault-Spieker J, Terveen L G, Hecht B. Avoiding the south side and the suburbs: The geography of mobile crowdsourcing markets. Proceedings of the 18th ACM Conference on Com-puter Supported Cooperative Work & Social Computing. ACM, 2015: 265-275.

[42] Ikeda K, Hoashi K. Crowdsourcing GO: Effect of Worker Situation on Mobile Crowdsourcing Performance. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM, 2017: 1142-1153.

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



Feng Wang is a master student at School of Electronic Engineering and Computer Science, Peking University, China. His research interest is mobile crowd sensing.

Yasha Wang received his Ph.D. degree in Northeastern University, Shenyang, China, in 2003. He is a professor and associate director of National Research & Engineering Center of Software Engineering in 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 and 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 is a professor at Peking University, China, and Télécom SudParis, France. He obtained his Ph.D from the University of Rome La Sapienza, Italy, in 1996. 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 forACM Transactions on Intelligent Systems and Technology,IEEE Transactions on Big Data, and others.

Brain Lim is currently an assistant professor in National University of Singapore. He obtained his Ph.D. from Carnegie Mellon University at Pittsburgh, US, in 2012. He has published a number of papers in prestigious conferences and journals, such as CHI, UbiComp, GROUP and etc. His research interests include ubiquitous computing and human-machine systems.

Leye Wang is currently a postdoctoral research fellow in Hong Kong University of Science and Technology. He obtained his Ph.D. from Institut Mines-Telecom/Telecom SudParis and Universite Pierre et Marie Curie, France, in 2016. He received his M.Sc. and B.Sc. in computer science from Peking University, China. His research interests include mobile crowdsensing and social networks, and intelligent transportation systems.