


# A location-constrained crowdsensing task allocation method for improving user satisfaction

International Journal of Distributed  
Sensor Networks  
2019, Vol. 15(10)  
© The Author(s) 2019  
DOI: 10.1177/1550147719883987  
[journals.sagepub.com/home/dsn](http://journals.sagepub.com/home/dsn)  


Huihui Chen<sup>1</sup> , Bin Guo<sup>2</sup>, Zhiwen Yu<sup>2</sup> and Liming Chen<sup>3</sup>

## Abstract

Mobile crowdsensing is a special data collection manner which collects data by smart phones taken by people every day. It is essential to pick suitable workers for different outdoor tasks. Constrained by participants' locations and their daily travel rules, they are likely to accomplish light outdoor tasks by their way without extra detours. Previous researchers found that people prefer to accomplish a certain number of tasks at a time; thus, we focus on assigning light outdoor tasks to workers by considering two optimization objectives, including (1) maximizing the ratio of light tasks for different workers and (2) maximizing the worker's satisfaction on assigned tasks. This task allocation problem is a non-deterministic polynomial-time-hard due to two reasons, that is, tasks and workers are many-to-many relationships and workers move from different places to different places. Considering both optimization objectives, we design the global-optimizing task allocation algorithm, which greedily selects the most appropriate participant until either no participant can be chosen or no tasks can be assigned. For the purpose of emulating real scenarios, different scales of maps, tasks, and workers are simulated to evaluate algorithms. Experimental results verify that the proposed global-optimizing method outperforms baselines on both maximization objectives.

## Keywords

Mobile crowdsensing, task allocation, location-constrained task

Date received: 12 May 2019; accepted: 29 September 2019

Handling Editor: Salvatore Distefano

## Introduction

The smart phones, wearable devices, and mobile social networking techniques have made mobile crowdsensing (MCS) and computing (MCSC)<sup>1</sup> a popular research area in recent years. MCSC leverages cross-space, heterogeneous crowdsourced data for large-scale sensing and computing. Participatory MCS is considered as an effective way to intensively collect professional data for data requester through publishing tasks and recruiting workers. A participatory task is a data collection campaign composed of active sensing actions, requiring a certain degree of human collaboration from workers and their direct involvement, such as going to a place in the city and taking a picture, answering a survey, and tagging a place.<sup>2,3</sup> Location-constrained tasks

usually require workers to be at the outdoor scene,<sup>4-8</sup> but there are not always participants who are at the task's place at the time of need and would like to accept tasks. Therefore, we have to ask people to go to places specified in tasks and accomplish different data-sampling tasks, which is usually referred to as location-constrained multi-task allocation.

<sup>1</sup>Foshan University, Foshan, China

<sup>2</sup>Northwestern Polytechnical University, Xi'an, China

<sup>3</sup>De Montfort University, Leicester, UK

### Corresponding author:

Huihui Chen, Foshan University, 8 Jiangwan 1st Road, Foshan 528225, China.

Email: [ddchh@163.com](mailto:ddchh@163.com)



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<http://www.creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work

without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

The task provider specifies the 2W2H (including Where, When, How and How many) requirement of the MCS task<sup>4,9</sup> on the MCS platform and expects plenty of high-quality data.<sup>2</sup> There are two basic methods for task allocation, that is, push-based task allocation and pull-based task allocation. The former method uses MCS platform to broadcast tasks to participants and some of them will accept to take the tasks. The latter method allows participants to query tasks and select tasks by their own on the MCS task platform. In this article, we marry the advantages of these two methods for the light task allocation. First, participants use the pull-based method to share their current locations and submit a task query (including the destination, her or his expected task number, and the rough duration of MCS working). Then, the MCS platform computes personalized suitable tasks for those participants according to their queries by using proposed task allocation methods in this article and uses the push-based method to publish suitable tasks to participants who are chosen to be workers. The task allocation component of the MCS platform is a black box for both workers and task providers, and reasonable task allocation results will attract more task providers and more participants, which is very important for the healthy MCS system.

There are two challenges to address the location-constrained multi-task allocation problem, including (1) people have different outdoor schedules and tasks must be attractive for them to become workers; (2) workers constantly change their locations during performing different tasks, hence the task allocation process should be based on a dynamic worker set. There are two types of tasks for workers, namely, light tasks and heavy tasks. A *light task* refers to a task that a worker can perform it without making a detour. Otherwise, it is a *heavy task* if a worker has to make detour to perform the task. *A task can be one worker's light task but another worker's heavy task.* To address the first challenge, we lower workers' labor intensity by only *allocating light tasks* to workers and ensure them expected rewards by *assigning multiple tasks expected by them*,<sup>10-12</sup> which also improve their satisfaction. Although workers' locations continually change, tasks' locations are stationary. As a result, to address the second challenge, we utilize the stationary locations of tasks to create the light task relationship matrix for every worker candidate and globally optimize workers' tasks with this matrix.

In this article, the proposed framework addresses the trade-off between the task allocation ratio and the worker satisfaction, and our main contributions consist of the following:

We analyze the light task allocation problem of MCS and propose two optimization objectives, that

is, (1) maximizing the mean satisfaction degree of workers and (2) maximizing the ratio of allocated tasks.

We design the personal light task relationship matrix which is used for two local-optimizing task allocation strategies, namely, (1) worker-first: searching maximum light tasks for each worker and (2) task-first: searching maximum suitable workers for each task.

We propose a global-optimizing task allocation algorithm, and the participant who can take the maximum suitable tasks will be prior chosen to be a worker. In this way, tasks are allocated as many as possible and workers will get their expected number of tasks leading to satisfied experience with task allocation.

We conduct experiments with different sizes of data sets and prove that both the ratio of allocated tasks and the mean satisfaction of workers can be increased by proposed optimal algorithms. Comparing to baseline and local-optimizing task allocation methods, the global optimizing algorithm shows higher evaluation results in both the task allocation ratio and the satisfaction degree of workers.

The remaining of this article is organized as follows. Section "Related work" introduces related work. Section "Problem formulation" formulates the light task allocation problem and defines two optimization objectives. Section "Algorithms" describes four task allocation algorithms, including three local-optimizing allocation algorithms and one global-optimizing task allocation algorithm. Section "Evaluation" presents experiments and discuss the results, followed by conclusions in section "Conclusion."

## Related work

There has been a significant amount of MCS applications in regard to utilizing user-contributed or crowd-sourced data. Different domains require different human inputs and different sensor readings, thus hybrid multiple-task management platforms are developed, where different tasks may have different temporal and spatial requirements. A task may be taken by one or multiple workers, depending on the application domain and the task requirements. Similarly, a worker may take multiple tasks. Participants' mobility is always considered by different location-constrained task allocation methods. In order to economically assign the right tasks to the right participant, some work pay attention to predicting and utilizing worker mobility,<sup>13-17</sup> while some works focus on high-performance task allocation methods.<sup>15,18,19</sup>

### Location-based MCS tasks

Location-constrained tasks exist in the physical world, and people are rewarded for their labors to go to a specific place and accomplish the tasks.<sup>3,4,15,20</sup> Two kinds of rewards are commonly seen. First, people are morally encouraged after they accomplish tasks, for instance, people upload photos of sidewalk issues through SeeClickFix App (<https://seeclickfix.com/>, accessed 2 November 2017). Second, people are paid by data consumers. For instance, FlierMeet paid money to participants for their contributed photos of fliers.<sup>3</sup> No matter which incentive mechanism is used, saving the cost of data collection is always considered as a key problem of every MCS application.

A lot of applications need the location information of the data. Different location accuracy are used for data sampling by different MCS applications. According to the location accuracy requirement, a spatial area is usually divided into a lot of grids and the data will be valid if the participant gets them in the task-specified grid, named grid-based sampling.

Some applications require low-accuracy location information, and then, the grid size will be a little larger, for example, 0.01–1 km<sup>2</sup>. If the grid is very large, then any people in the same grid can be chosen to perform data-sampling task; therefore, these applications can recruit a large number of participants with the low labor cost because most participants can stay where they are and perform task, such as most MCS-based noisy monitoring systems and air quality monitoring systems.<sup>14</sup> The task allocation of these applications will not request high-precision location information of users, for example, global positioning system (GPS) points, but only use the low-precision and low-power-consumption location information, for example, cell tower localization. For a multiple-task allocation system, the participant recruitment is influenced by the grid size. First, if the grid are very large, then a lot of participants do not need to go to different grids and they can perform many different tasks in the same grid or some closed grids. Second, participants are unevenly distributed and a larger grid can easily contain participants. Third, packaged multi-tasks are more attractive for participants than single task.<sup>10</sup> In this article, we adjust the grid size and investigate the consequent effect on the task allocation.

Sometimes, for the purpose of protecting user's privacy, the task allocation system will not require users' high-precision locations, so the task allocation server uses the rough location of participants to allocate tasks. Some location-constrained applications must recruit more participants to collect adequate data. In order to lower the cost of recruiting participants, MCS-based applications normally leverage highly effective task allocation algorithms. Task allocation and completion

rates are severely affected by limited resources, so Liu et al.<sup>15</sup> investigated two situations, that is, scarce participants and adequate participants, and proposed two optimal task allocation algorithms based on the Minimum Cost Maximum Flow theory. The optimization objective of Liu et al.<sup>15</sup> and Guo et al.<sup>17</sup> is to minimize the total distance that workers go to different tasks' locations and maximize the number of accomplished tasks.

People are inclined to take tasks that can be easily performed on their ways to somewhere, so several studies leverage human mobility prediction to assign tasks.<sup>13–15,21</sup> Wang et al.<sup>13</sup> considered the original task allocation problem as the form of matching discovered mobility patterns with MCS task sequences, with the goal of minimizing the sensing cost. Ji et al.<sup>14</sup> also considered human mobility and proposed a urban MCS framework that maximizes the coverage of collected data in a spatio-temporal space, based on human mobility of participants recruited by a given budget. These works leveraged previous mobility data of participants to predict the following routes of them, which protect participants' privacy but need to recruit redundant workers to obtain adequate samples due to some failure task assignments. Zhao et al.<sup>6</sup> investigated the task assignment of spatial crowdsourcing under destination-aware task assignment and aimed at finding an optimal assignment of tasks to workers such that the total number of accomplished tasks is maximized. In this article, we also consider human mobility. We require participants' exact planned routes with either low or high location accuracy and their expected task numbers in order to precisely allocate tasks to the most suitable participants who will get expected numbers of tasks without making a detour.

### Location-constrained task allocation

The location constraints of tasks are set by most MCS systems, which guide recent studies to consider the spatial relationships between participants and tasks and maximize the spatial coverage of participants.<sup>11,19,22</sup> Reddy et al.<sup>23</sup> proposed a greedy participant selection method to improve coverage of a limited geographic scope. CrowdRecruiter<sup>22</sup> selects the near-minimal set of participants, which meets coverage ratio requirement in each sensing cycle of the task. It predicts the call and coverage probability of each mobile user based on historical records and then computes the joint coverage probability of multiple users as a combined set. No matter what constraints are given, the common objective of some studies is to maximizing the spatial coverage of selected participants.<sup>19,22,20</sup> Time-to-live of tasks were also focused in some work, such as minimizing the total time cost of performing tasks through utilizing the social network<sup>11,24</sup> and minimizing the delay of

performing tasks to ensure the data quality through utilizing the participant's start position and the task's valid duration. The task allocation is always related to the incentive strategy and the budget, so research has been undertaken to limit or minimize the budget of participant recruitment.<sup>25</sup> In this article, tasks' locations are geographic points and their distribution has no regular patterns, so the spatial coverage is actually the ratio of allocated tasks. On one hand, considering lowering the incentive budget, we lower the cost through performing light tasks. On the other hand, considering satisfying participants, tasks are assigned as many as possible to the same participant. Therefore, different from recent work, our proposed methods will satisfy two optimization objectives for location-constrained MCS task allocation.

## Problem formulation

This section presents the location-constrained light task allocation problem for MCS on the multiple workers and multiple-task platform.

### The multi-task allocation problem

Location-constrained tasks usually require workers to be at the scene, but it is impossible to always recruit workers at the exact venue defined by the task. Therefore, we have to ask workers to go to some specified places to take pictures, and this process is called task assignment.

In order to introduce the problem, two kinds of participants are defined. If a participant shares her location and submits a task query, then the participant becomes a *worker candidate* (*candidate* for short). Consequently, if a worker candidate gets works, then the worker candidate becomes a *worker*.

A worker (or a candidate) is denoted by  $w$  and has three parameters  $\langle e, s, d \rangle$ .  $W$  denotes the workers' set. For the  $j$ th worker,  $e_j$  denotes the expected task number of  $w_j$ .  $s_j$  denotes the start point (i.e. the current location) of  $w_j$ .  $d_j$  denotes the destination where  $w_j$  is originally going.

Each worker's expected number of tasks (i.e.  $e$ ) is set by himself/herself or is calculated according to the historical task accomplishment records, which reflect workers' task accomplishment expectation and limitation.

A task is denoted by a three-tuple  $t_i = \langle l_i, r_i, H_i \rangle$ .  $l_i$  denotes the task location,  $r_i$  denotes the required number of workers, and  $H_i$  is a set that denotes recruited workers of this task. Some MCS applications may require data sampled at different places, and we consider that this application will publish multiple tasks located at different places in this article.

**Table I.** Frequently used notations.

Notation	Explanation
$W$	The worker and candidate set, $w_j \in W$ .
$B_j$	The task set of the worker/candidate $w_j$ .
$s_j$	The start or current location of $w_j$ .
$d_j$	The destination of $w_j$ .
$e_j$	The maximum number of tasks taken by $w_j$ , $ B_j  \leq e_j$ .
$T$	The task set, $t_i \in T$ .
$H_i$	The worker set of the task $t_i$ .
$l_i$	The GPS coordinates of the sensing task $t_i$ .
$r_i$	The maximum number of workers required by $t_i$ , $ H_i  \leq r_i$ .
$\mathbf{A}$	The task allocation result, $a_{i,j} \in \mathbf{A}$ .
$a_{i,j}$	$a_{i,j} = 1$ means $t_i$ is assigned to $w_j$ , otherwise, $a_{i,j} = 0$ .

GPS: global positioning system.

Given a candidate set  $W$  and a task set  $T$ , the matrix  $\mathbf{A} = a_{i,j}(|T| \times |W|)$  denotes the task allocation result, where  $a_{i,j} = 1$  means that the task  $t_i \in T$  is allocated to the worker  $w_j \in W$ . Thus, tasks assigned to the worker  $w_j$  compose  $B_j = \{t_m | a_{m,j} = 1, t_m \in T\}$  and workers recruited by the task  $t_i$  compose  $H_i = \{w_m | a_{i,m} = 1, w_m \in W\}$ . Therefore, the process of task allocation is to compute the matrix  $\mathbf{A}$ .

The general task assignment problem can be formulated to an optimization problem as shown in equation (1), where the extra movement (i.e. detour) is minimized

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmin}} \left\{ \sum_{j=1}^{|W|} (\mu_j - \varphi_j) \right\} \quad (1)$$

s.t.

$$\forall w_j \in W (|B_j| \leq e_j), \forall t_i \in T (|H_i| = r_i)$$

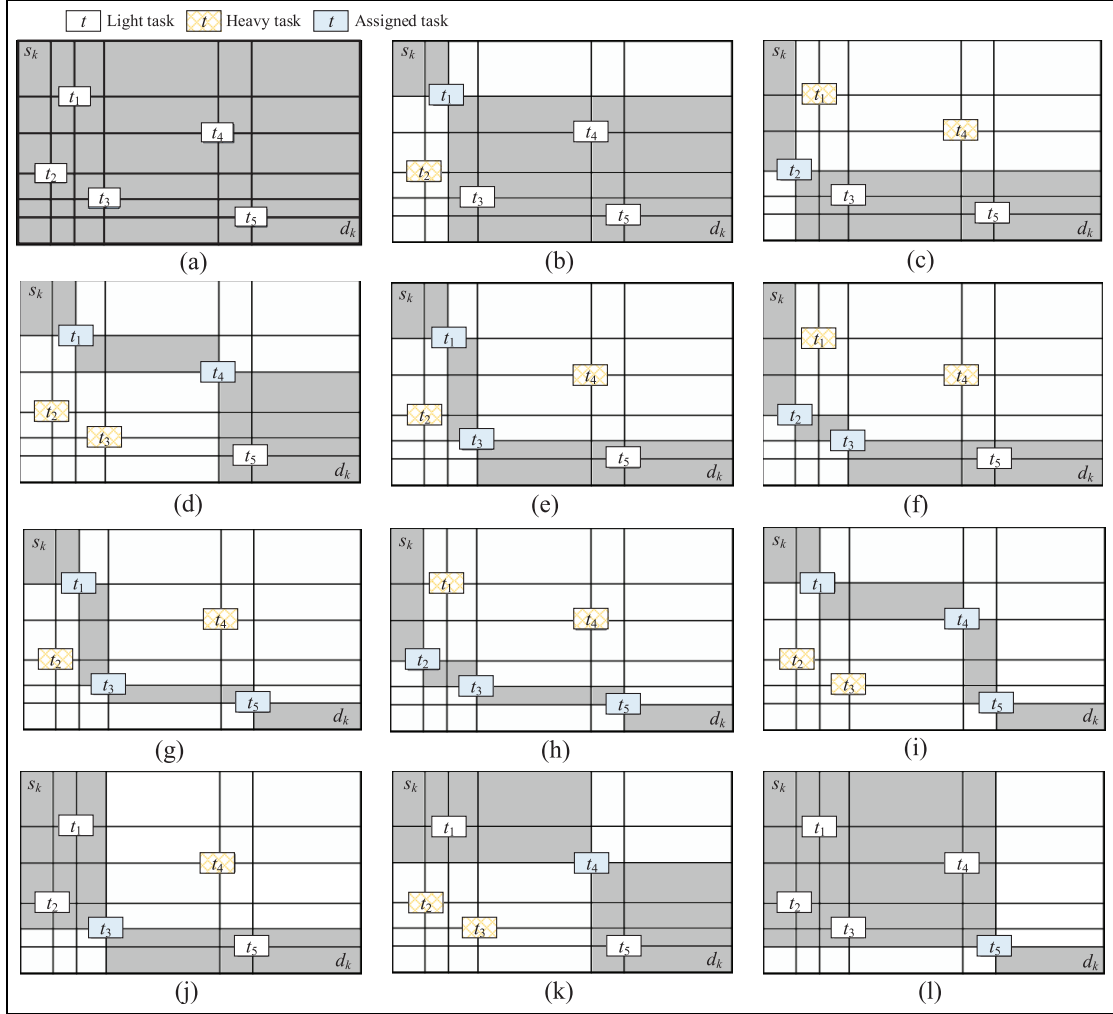
where  $\varphi_j$  denotes the distance of the  $j$ th worker's original route and  $\mu_j$  denotes the minimal distance for finishing all tasks.

The problem in equation (1) is the multi-task assignment problem and is non-deterministic polynomial-time (NP)-hard. The frequently used notations are shown in Table 1.

### Fast light task searching method

Searching the shortest route is a NP-hard problem, so the *Manhattan distance* is used to explain the fast light tasks finding method as follows.

*Task-performing route:* At the beginning, the worker  $w_j$  plans to go to  $d_j$  from  $s_j$ . If this worker gets some tasks, then she must visit different places and gets data, for example, taking a photo. Assume that the shortest path for accomplishing tasks in  $B_j$  is



**Figure 1.** The different task allocation results and the corresponding no-detour working areas: (a)  $|B_k| = \{\}$ , (b)  $|B_k| = \{t_1\}$ , (c)  $|B_k| = \{t_2\}$ , (d)  $|B_k| = \{t_1, t_4\}$ , (e)  $|B_k| = \{t_1, t_3\}$ , (f)  $|B_k| = \{t_2, t_3\}$ , (g)  $|B_k| = \{t_1, t_3, t_5\}$ , (h)  $|B_k| = \{t_2, t_3, t_5\}$ , (i)  $|B_k| = \{t_1, t_4, t_5\}$ , (j)  $|B_k| = \{t_3\}$ , (k)  $|B_k| = \{t_4\}$ , and (l)  $|B_k| = \{t_4\}$ .

denoted by  $Q_j = \{q_{j,1}, \dots\}$ , where  $q_{j,m}$  is the identifier of a task, which means that the worker  $w_j$  will move from  $s_j$  to  $l_{q_{j,1}}, l_{q_{j,2}}, \dots, l_{q_{j,|Q_j|}}, d_j$  in sequence.

**Task rectangles:** Key points (denoted by  $p_i$ ) of a worker's task-performing route include the starting point, all locations of the tasks to be taken, and the destination point. Based on two consecutive key points  $\{p_i, p_j\}$ , a rectangle whose opposite vertexes are  $p_i$  and  $p_j$  and edges are parallel to either east-west axis or south-north axis can be created. This rectangle is denoted by  $\Pi(p_i, p_j)$  and is called task rectangles. Task rectangles of one worker are linked by task location points. For example, the task-performing route is  $p_1 - p_2 - p_3 - p_4$ , and then, task rectangles are  $\Pi(p_1, p_2)$ ,  $\Pi(p_2, p_3)$ , and  $\Pi(p_3, p_4)$ .

**No-detour working area:** The area will be called a worker's no-detour working area if there is always a no-detour travel route shorter than the Manhattan distant from the starting point to the destination. According to the constraint of a light task, we can consider any task covered by the no-detour working area as a light task. According to the definition of the task rectangle, we can find that the no-detouring area consists of all task rectangles.

In this article, we only assign light tasks to worker, so the worker is supposed to be able to always find the shortest route in each task rectangle; therefore, task rectangles of a worker will not overlap to each other. As shown in Figure 1, the dark areas present no-detour working areas (area for short). At beginning, the area

is composed of the task rectangle  $\Pi(s_k, d_k)$  (see Figure 1(a)), so any covered task can be assigned to  $w_k$ , that is,  $\{t_1, t_2, t_3, t_4, t_5\}$ . Since the worker must visit all task places, once he or she gets a new task, the area must change. For example, as shown in Figure 1(b), the area will be changed to  $\Pi(s_k, l_1)$  and  $\Pi(l_1, d_k)$  after the task  $t_1$  is assigned. It is easy to see whether the worker takes different tasks, then the area will be different (see Figure 1(c), (k), and (l)), and the light task set changes along with the task allocation process. After task  $t_1$  is assigned to the worker, the light task set changes to  $\{t_3, t_4, t_5\}$  (see Figure 1(b)), and after the task  $t_4$  is assigned, it changes to  $\{t_5\}$  (see Figure 1(d)). We continuously select light tasks for different workers until the task number reaches the expected number of each worker. Figure 1 shows differences in both no-detour working areas and task allocation results if using different task allocation sequences. For example, Figure 1(b), (e), and (g) is for the sequence of selecting tasks  $\{t_1, t_3, t_5\}$ , and Figure 1(c), (f), and (h) for the sequence  $\{t_2, t_3, t_5\}$ , and Figure 1(b), (d), and (i) for the sequence  $\{t_1, t_4, t_5\}$ . As the light task set changes and there are multiple task selection solutions for each worker, this will lead to a large number of solutions for multiple workers, which brings challenges for computing the optimal task allocation result.

### Optimization objectives of task allocation

Location-constrained light task allocation (light task allocation for short here after) is a specific problem of the multi-task allocation problem. It only assigns each worker light tasks. There are two objectives of light task allocation. On one hand, since using light task allocation will not bring extra distance to workers, we do not need to consider minimizing the distance of trips but focus on how to *allocate more tasks*. On the other hand, in order to *improve workers' experiences*, tasks should be intensively assigned to workers, so the number of assigned tasks should be as much close to a workers' expected number as possible.

Musthag and Ganesan<sup>26</sup> revealed that a small fraction of agents (<10% of all agents), whom they referred to as super-agents, performed more than 80% of the tasks and earn more than 80% of the total earnings. Thus, super-agents' experiences are very important. The *satisfaction degree* of  $j$ th worker is reflected by the degree of  $j$ th worker's task number reaching his or her expected task number, denoted by  $f_j$  calculated by equation (2). The two objectives of the task allocation can be formally defined as (1) *to maximize the task allocation ratio* (written in short as MAX-I) and (2) *to minimize the difference between the expected number and the real number of assigned tasks, that is, to maximize the satisfaction degree* (written in short as MAX-II)

$$f_j = \frac{|B_j|}{e_j} \quad (2)$$

Given a candidate set  $W$  and a task set  $T$ , the matrix  $\mathbf{A} = a_{i,j}(|T| \times |W|)$  denotes the task allocation result, and  $a_{i,j}$  is valued as equation (3)

$$\begin{cases} a_{i,j} = 1, & \text{if } t_i \text{ is assigned to } w_j \\ a_{i,j} = 0, & \text{otherwise.} \end{cases} \quad (3)$$

The objective is formulated as equation (4)

$$\text{Maximize } \left\{ \frac{\sum_{i=1}^{|T|} \sum_{j=1}^{|W|} a_{i,j}}{|T| \times |W|} \times \frac{\sum_{j=1}^{|W|} f_j}{|W'|} \right\} \quad (4)$$

s.t.

$$\sum_{j=1}^{|W|} a_{i,j} \leq r_i, \quad \sum_{i=1}^{|T|} a_{i,j} \leq e_j$$

Notice that if  $|W'| < |W|$ , then  $|W'|$  in different work allocation results may be different, and two objectives, that is MAX-I and MAX-II, might conflict. For example, a worker Tom requires three tasks, and only one task can be assigned to him as well as only Tom can be chosen as this task's worker. If Tom is chosen, then the number of allocated tasks increases but the mean satisfaction degree decreases. Therefore, we focus on maximizing the satisfaction degree meanwhile avoiding not to lower the task allocation rate.

Location-constrained light task allocation problem can be considered as a multi-dimensional and dynamic knapsack problem. A worker's original route can be regarded as the capacity of the bag and tasks are items. Since a worker must go to the task places, items' weights change. If items in the bag are different, then weights of outside items are different. Therefore, the light task allocation problem is NP-hard.<sup>27</sup> In the next section, we describe three local-optimizing task allocation methods and one global-optimizing task allocation method based on the greedy algorithm to address this problem.

### Algorithms

In the case of there are multiple tasks and multiple workers, then two priority strategies can be used, including the *worker-first strategy* and the *task-first strategy*. The worker-first strategy will assign tasks to each worker based on the workers' requests and preferences, while the task-first strategy will select workers for each task based on a task's requirements. No matter which strategy is used, if only considering one worker or one task during each selection round, the result could be locally optimal. To address multiple

workers and multiple tasks scenarios, we propose a global-optimizing method in this section.

### Local-optimizing task allocation methods

**Worker-first local-optimizing algorithm.** Using an enumerative method, we can find the optimal solution. Before introducing the enumerative method, we define the directed no-detouring relationship, dubbed as no-detouring subsequence, between task pairs. Given a worker  $w_i$  and her two light tasks  $t_j$  and  $t_k$ , if  $\varphi(s_i, l_j, d_i) = \varphi(s_i, l_j, l_k, d_i)$  where the function  $\varphi$  calculates the distance of a route, then  $t_k$  is  $t_j$ 's no-detouring subsequence task.

Each task can have zero or multiple no-detouring subsequence task. Given the available task set  $T$ , the process of finding tasks for a worker  $w_k$  based on worker-first strategy consists of (1) the enumerative forest generation and (2) the task allocation. The enumerative forest generation algorithm is shown in Algorithm 1, which utilizes the no-detour subsequence relationships between tasks and the depth-first searching to enumerate all task allocation instances. First, we create a directed graph based on no-detouring subsequence relationships between tasks. Second, through deep-first traversing, a forest can be generated based on this graph.

An instance of the enumerative forest generation is shown in Figure 2. As shown in Figure 2(a), for the  $k$ th worker, the initial task rectangle is  $\Pi(s_k, d_k)$  and all tasks covered by this rectangle are initially this worker's light tasks, for example,  $\{t_1, t_2, \dots, t_5\}$ . The directed graph based on tasks' no-detouring subsequence relationships is shown in Figure 2(b). Figure 2(c) shows the traversing results, that is, a forest.

---

#### Algorithm 1. Enumerate forest method.

---

```

1: //Creating a directed graph. All light tasks of  $w_k$  in  $T$  are vertexes
   and an edge from  $t_i$  to  $t_j$  means that  $t_j$  is  $t_i$ 's no-detouring task.
2:  $V \leftarrow$  All light tasks of  $w_k$ ;
3:  $E \leftarrow$  No detour relationships between elements in  $V$ ;
4:  $G \leftarrow \langle V, E \rangle$ ; //Create a directed graph.
5: //Traversing all vertexes and generating a forest.
6:  $R \leftarrow$  Finding all zero in-degree node in  $G$ ;
7: for each  $r \in R$  do
8:   Create a tree  $T_r$  whose root node is  $r$ ;
9:    $T_r \leftarrow$  DFS( $r, T_r, G$ ); //See Algorithm 2.
10: end for

```

---

If there is a branch whose length is larger than or equal to  $e_k$  in the enumerative forest,  $e_k$  nodes (denoting  $e_k$  tasks) on this branch compose an optimal task set of the worker  $w_k$ . Otherwise, if all branches are shorter

---

#### Algorithm 2. Function DFS ( $r, T_r, G$ ).

---

```

Require:  $r, T_r, G$ ;
Ensure:  $T_r$ ;
1: //Depth-First Searching and generating a tree.
2:  $C \leftarrow$  All next nodes of  $r$  in the directed graph  $G$ ;
3: for each  $c \in C$  do
4:   Create a new branch ( $r \rightarrow c$ ) on the tree  $T_r$ ;
5:    $T_r \leftarrow$  DFS( $c, T_r, G$ );
6: end for
7: return  $T_r$ .

```

---

than  $e_k$ , then nodes (i.e. tasks) on the longest branch compose the optimal task set. Meanwhile, from top to down, tree nodes compose the task-performing route. As shown in Figure 2(c), if  $e_k \geq 3$ , then three task sets  $\{t_2, t_3, t_5\}$ ,  $\{t_1, t_3, t_5\}$ , and  $\{t_1, t_4, t_5\}$  can be the optimal task set for  $w_k$ .

Cheng et al.<sup>20</sup> proposed a greedy algorithm, which iteratively assigns workers to spatial tasks that can always achieve high ranks, which is a worker-first algorithm. In this article, given a candidate set, the worker-first local-optimizing (WF-LO) algorithm generates the enumerative forest of one candidate at one time and assign maximum tasks to this candidate who then becomes a worker. Candidates are considered and assigned tasks one by one. Here, the optimal task set is only optimal for one worker. As each task requires limited number of workers, tasks assigned to one worker will influence another's task assignment.

**Native task-first algorithm (TF-N).** The location of a task is fixed, any worker whose no-detour working area covers this task can be chosen for this task, called a valuable candidate. The TF-N algorithm selects suitable workers for one task at a time.

Given a task  $t_k$ , assuming that the present task sequence (the route has been planned according to tasks' locations) of the worker  $w_i$  is  $q_{i,1} - q_{i,2} - \dots$ , the rule to judge whether the task  $t_k$  can be assigned to this worker or not is that whether the point  $l_k$  is in the no-detour working area that computed according to  $s_i - l_{q_{i,1}} - l_{q_{i,2}} - \dots - d_i$ . If the point  $l_k$  is in the rectangle formed by  $l_{q_{i,j}}$  and  $l_{q_{i,j+1}}$ , then task  $t_k$  is inserted into the task sequence between  $q_{i,j}$  and  $q_{i,j+1}$ . The task-performing route will be  $s_i - l_{q_{i,1}} - l_{q_{i,2}} - \dots - q_{i,j} - l_k - q_{i,j+1} - d_i$ . The worker selection of one task might influence another task's valuable worker set. As shown in Figure 1(b), tasks  $t_3, t_4$ , and  $t_5$  can select the worker  $w_k$ . If  $w_k$  is chosen by  $t_3$  (see Figure 1(e)), then  $t_4$  cannot choose  $w_k$  any more. Considering the objective MAX-II, we propose the task-first local-optimizing (TF-LO) algorithm in the following.

**TF-LO.** Given a task  $t_i$ , TF-LO algorithm finds all workers who can take this task and then chooses workers whose allocated task number nears their expected task numbers. This algorithm considers the second objective MAX-II. The detailed process is shown in Algorithm 3. For each worker  $w_j$  who is able to take the task  $t_i$ , we compute the satisfied ratio  $((|B_j| + 1)/e_j)$  by assuming that  $t_i$  is assigned to this worker. All workers' satisfied ratios are sorted in descending order, and the top  $r_i - |H_i|$  workers are recruited by task  $t_i$ .

Because the no-detour working area changes, only one task can select this worker and then refresh the no-detour working area for the next round of worker selection. Therefore, although the task location is stable to simplify the task allocation, either using worker-first strategy or using task-first strategy for multi-task allocation must consider changes of tasks' worker numbers, workers' task numbers, and worker's locations during the task allocation process, which is a very complex problem. In the following, we propose a greedy-based task allocation algorithm with global optimization.

---

**Algorithm 3.** Task-first global-optimizing algorithm.

---

```

1: for each  $t_i$  in  $T$  do
2:   for each  $w_j$  in  $W$  do
3:      $c_j \leftarrow 0$ ;
4:     if  $|B_j| < e_j$  and  $t_i$  is a light task for  $w_j$  then
5:        $c_j \leftarrow \frac{|B_j| + 1}{e_j}$ ;
6:     end if
7:   end for
8:   while  $|H_i| < r_i$  do
9:      $p \leftarrow$  The index of the largest element in  $\{c_1, \dots, c_{|W|}\}$ ;
10:     $H_i \leftarrow H_i + w_p$ ; //Task  $t_i$  is assigned to  $w_p$ .
11:     $B_p \leftarrow B_p + t_i$ ; //Task  $t_i$  recruits  $w_p$ .
12:     $c_p \leftarrow 0$ ;
13:  end while
14: end for

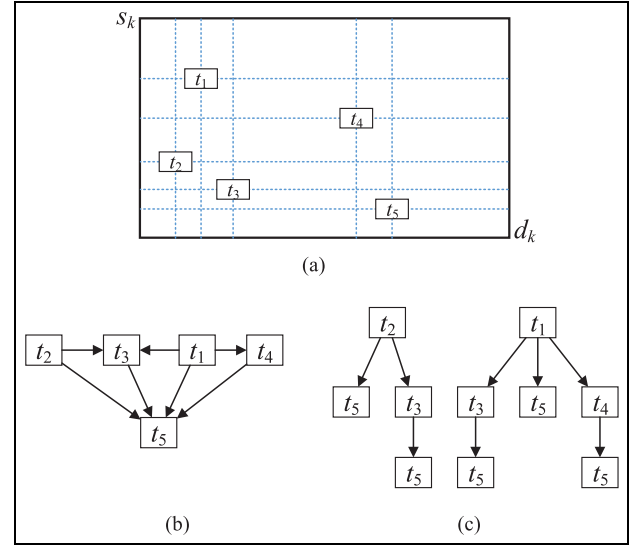
```

---

### Global-optimizing task allocation methods

**Light task packaging.** For one worker, we can find the optimal task package with the enumerative method, however, for multiple workers, it is hard to enumerate the optimal task allocation solution. Tasks that can be assigned to the same worker are considered as a task package. As shown in Figure 2(c), according to the enumerative forest and the task number parameter (i.e.  $e_i$ ), there are multiple task-packaging possibilities, for example,  $\{t_2, t_5\}$  and  $\{t_1, t_3, t_5\}$ .

Given  $n$  tasks  $T = \{t_1, t_2, \dots, t_n\}$  and  $m$  candidates  $W = \{w_1, w_2, \dots, w_m\}$ , a candidate's task package set is denoted by a matrix  $\hat{D}_i = \{D_{i,1}, D_{i,2}, \dots\}$ , where  $D_{i,j} = \{d_{i,j,k} | k = 1, 2, \dots, n\}$  is a  $n$ -dimensional row



**Figure 2.** The instance of no-touring task allocation: (a) light tasks covered by the initial task rectangle, (b) directed graph, and (c) forest.

vector.  $d_{i,j,k} = 1$  means task  $t_k$  is assigned to worker  $w_i$  in the  $j$ th task package. Since different workers have different trip plans, their task packages are also different. We select one task package from each worker's task packages and obtain a task allocation solution, which is denoted by  $K = (k_1, k_2, \dots, k_m)$  where  $1 \leq k_i \leq |\hat{D}_i|$ , and then, the corresponding task allocation result is denoted by  $P =$

$$P = \begin{bmatrix} D_{1,k_1} \\ D_{2,k_2} \\ \dots \\ D_{m,k_m} \end{bmatrix}.$$

For instance, assume that  $n = 4$ ,  $m = 3$ ,  $e_i = 2$  ( $i = 1, 2, 3$ ),  $r_j = 2$  ( $j = 1, 2, 3, 4$ ), and workers' task packages are  $D_1$ ,  $D_2$ , and  $D_3$ , respectively, as follows

$$D_1 = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}, D_2 = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

$$D_3 = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$



There are  $|D_1| \times |D_2| \times |D_3| = 168$  theoretical task allocation solutions. Subjecting to both worker number constraints of tasks (i.e.  $r_i$ ) and task number constraints of workers (i.e.  $e_j$ ), some task allocation solutions will be abandoned and the available task allocation solutions will be less. For example, either  $P_1$  or  $P_2$  is an optimal results of this instance

$$P_1 = \left[ \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ w_1 & 0 & 1 & 1 & 0 \\ w_2 & 0 & 0 & 1 & 1 \\ w_3 & 1 & 1 & 0 & 0 \end{array} \right], \quad K_1 = (5, 7, 4)$$

$$P_2 = \left[ \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ w_1 & 1 & 1 & 0 & 0 \\ w_2 & 0 & 0 & 1 & 1 \\ w_3 & 1 & 1 & 0 & 0 \end{array} \right], \quad K_2 = (6, 7, 4)$$

The number of theoretical task allocation solutions is  $\prod_{j=1}^m |\dot{D}_j|$  by using the enumeration method, which is inefficient to feedback candidates' task queries. Therefore, we use greedy-based algorithm to choose task packages, which is introduced in the following.

**Worker-first globally optimized algorithm.** For the purpose of both greedily assigning more tasks and satisfying most workers, we propose worker-first globally optimized (WF-GO) algorithm, which searches the most suitable worker rather than assigning the sequential candidate proper tasks like WF-LO.

WF-GO uses the backtracking to satisfy both MAX-I and MAX-II at the same time. First, all task packages of each worker is created and then sorted in descending order of package sizes (which refers to the task number in the package). Second, every candidate is assessed to determine their selection. In the  $i$ th assessment round, the task number limitation of a candidate  $w_k$  is  $e_k - i + 1$ , and the assessment judges whether the candidate has an available task package whose size is equal to the task number limitation. Here, a task package is available only if all tasks in this package are able to recruit this candidate based on the intermediate result of current task allocation. Finally, some candidates become workers in each round of candidate assessment. The worker-searching process will stop once either all tasks are allocated or all remaining candidates have no available task packages.

## Evaluation

### Experiment setup

**Datasets and simulations.** We use the dataset of individual Divvy bike sharing trips to imitate workers' trips, including the origin, destination, and timestamps for each trip (<https://data.cityofchicago.org/browse?q=divvy>, accessed 12 October 2018); 386,809 trips of

3 months are used to evaluate our methods. The map is divided into  $g \times g$  grid cells.

For the same task set and the same worker set, if an area is divided into different numbers of grids, the density of workers and also the density of tasks will be different. The *sensing coverage*, denoted by  $ssCovr$ , is used as an index to reflect the ratio of grids that can be visited by workers.  $ssCovr$  is computed by equation (5)

$$ssCovr = \frac{1}{|W|} \times \frac{\sum_{i=1}^{|W|} \varphi(d_i, s_i)}{g \times g} \quad (5)$$

$ssCovr$  can influence the task allocation evaluation. Experimental results shows that if  $g = 10$ ,  $ssCovr \approx 0.05$ ; if  $g = 20$ ,  $ssCovr \approx 0.03$ ; and if  $g = 30$ ,  $ssCovr \approx 0.02$ . Although workers' current locations and destinations are uniform random values, we find that  $ssCovr$  nearly remains the same for different worker sets when  $|W| \geq 1000$ . In order to avoid being interfered by those biased random numbers, each experiment uses 1000 simulated task sets. Statistical results presented in the following are obtained based on experimental results of these 1000 datasets.

**Baselines and metrics.** Greedy algorithms are usually used by researches to efficiently allocate tasks.<sup>20,27,28</sup> We provide the following baseline methods for comparative studies.

Naive Greedy Allocation—TF-N algorithm.<sup>28</sup> Much work without optimization.

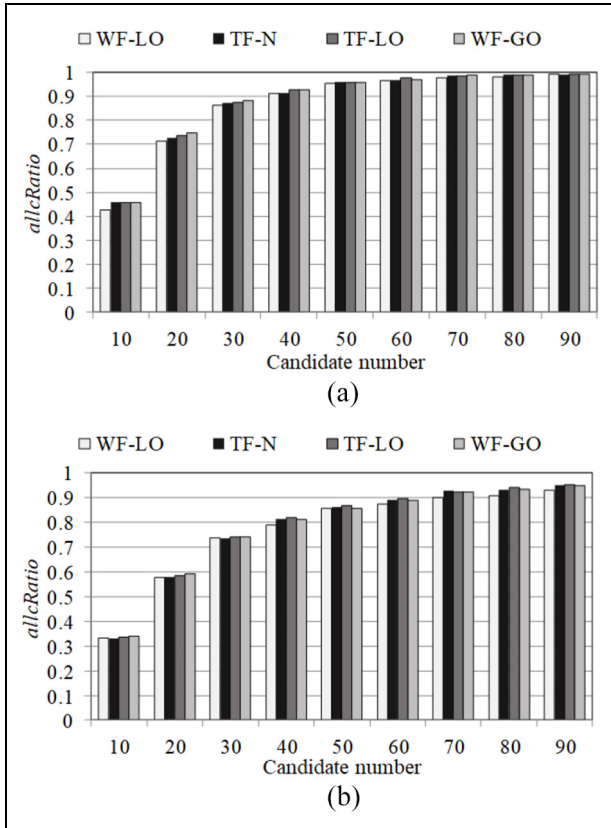
Local-optimizing Greedy Allocation—TF-LO algorithm and WF-LO algorithm.<sup>20,27</sup>

We use two evaluation metrics, namely, task allocation ratio and satisfaction degree. Given the task set  $T$  and the candidate set  $W$ ,  $allcRatio$  denotes the task allocation ratio and is computed by equation (6). The satisfaction degree of the worker is used to reflect whether the task number reaches this worker's expectation. The mean satisfaction degree (denoted by  $satDegree$ ) of a worker set was computed by equation (7), where the worker set (denoted by  $W'$ ) is a sub-set of the candidate set  $W$

$$allcRatio = \frac{\sum_{i=1}^{|T|} |H_i|}{\sum_{i=1}^{|T|} r_i} \quad (6)$$

$$satDegree = \frac{1}{|W'|} \times \sum_{j=1}^{|W'|} \frac{|B_j|}{e_j} \quad (7)$$

where  $W' \subseteq W$  and  $\forall w_j \in W' (|B_j| \geq 1)$ .

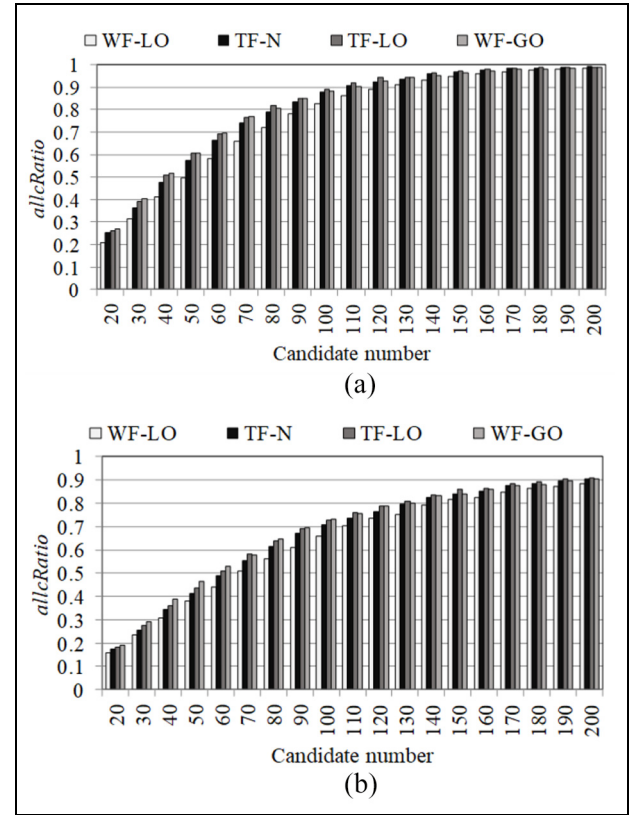


**Figure 3.** The experimental result of the task allocation ratio when  $|T| = 10$ ,  $\forall j(e_j = 3)$ , and  $\forall i(r_i = 3)$ . X-axis labels show that the number of workers gradually far exceeds task need: (a)  $g = 10$  and (b)  $g = 20$ .

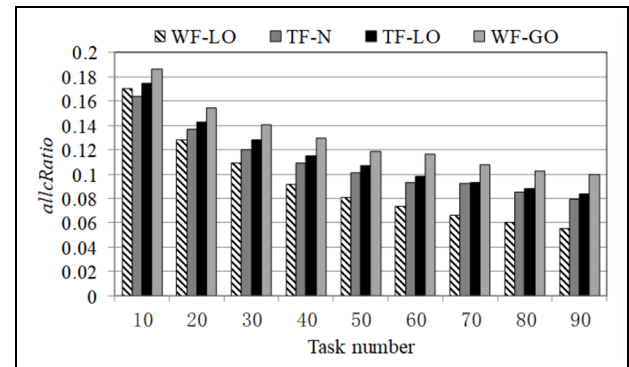
### Experimental results

**Task allocation ratio.** Figures 3 and 4 show the experimental results of the task allocation ratio, from which we can draw the following finding: (1) when the candidate number increases, the task allocation ratio also increases, which is in line with the common knowledge; (2) comparing to WF-LO, the task allocation ratio of using WF-GO increases and the increment shown in Figure 4 even reaches 10% when the worker number is small; (3) when  $g$  increases, the sensing coverage  $ssCovr$  decreases, thus the task allocation decreases; and (4) when the candidate number increases, the task allocation ratio differences among WF-LO, WF-GO, TF-N, and TF-LO become smaller.

According to experimental results and the above findings, we can conclude that if the candidate number is relatively small, using WF-GO can obtain higher task allocation ratio, and if there are adequate candidates, that is,  $\sum_{j=1}^{|W|} e_j \gg \sum_{i=1}^{|T|} r_i$ , using any method can obtain high task allocation ratio. For example, as shown in Figure 4(b), if worker number is less than 60,

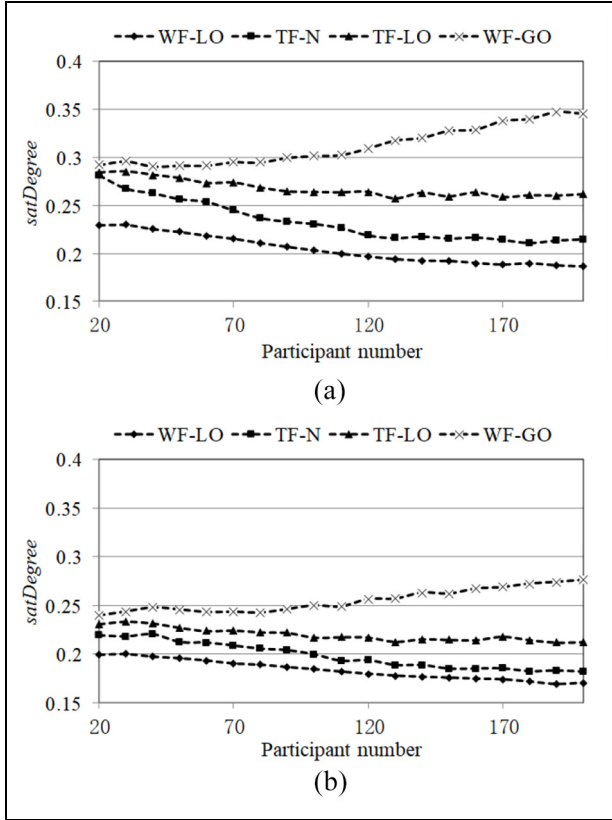


**Figure 4.** The experimental result of the task allocation ratio when  $|T| = 20$ ,  $\forall j(e_j = 10)$ , and  $\forall i(r_i = 10)$ . X-axis labels show that the number of workers gradually far exceeds tasks need: (a)  $g = 10$  and (b)  $g = 20$ .



**Figure 5.** The experimental result of the task allocation ratio when  $g = 50$ ,  $|W| = 20$ ,  $\forall j(e_j = 10)$ , and  $\forall i(r_i = 10)$ . X-axis labels show that the number of tasks is gradually higher than the number of tasks that workers can accomplish.

WF-GO is the best method. In order to evaluate the performance of the WF-GO method when the candidate number is small, we increase the task number but stabilize the worker number. The experimental result as shown in Figure 5 proved that more tasks can be



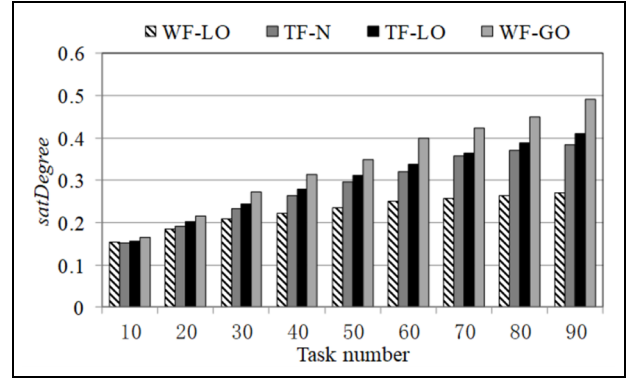
**Figure 6.** The experimental results of satisfaction degrees when  $|T| = 20$ ,  $\forall j(e_j = 10)$ , and  $\forall i(r_i = 10)$ . X-axis shows the number of workers gradually far exceeds tasks need: (a)  $g = 10$  and (b)  $g = 20$ .

allocated by the WF-GO method through optimally utilizing the limited candidate resource.

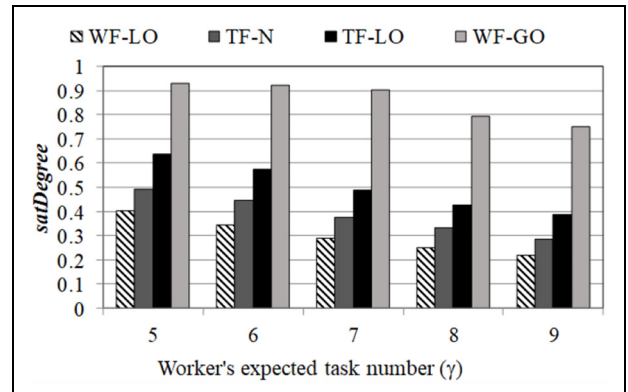
We also evaluate the proposed methods for maximizing the satisfaction degrees of workers, which is presented in the following.

**Satisfaction degree.** Constrained by no-detouring, candidates cannot be chosen to be workers if their routes do not pass any task's location. Given the same candidate set and the same task set, the smaller worker number means that the task is centrally allocated as well as the task number is closer to the worker's expectation.

We vary both the worker number and the task number, respectively, for the purpose of simulating situations of either adequate candidates or scarce candidates. Figures 6 and 7 show the experimental results of the satisfaction degree evaluation. As can be seen in Figure 6, when the participant number increases, the satisfaction degree of WF-GO increases. When more candidates are available, through using WF-GO, we globally select the most suitable candidate one by one, thus both the satisfaction degree (see Figure 6) and the task allocation ratio (see Figure 4)



**Figure 7.** The experimental results of satisfaction degrees when  $g = 50$ ,  $|W| = 20$ ,  $\forall j(e_j = 10)$ , and  $\forall i(r_i = 10)$ . X-axis labels show tasks are gradually more than that workers can accomplish.



**Figure 8.** The satisfaction degree comparison under adjusting the worker's expected task number when  $g = 50$ ,  $|W| = 10,000$ ,  $|T| = 100$ , and  $\forall j(e_j = 20)$ . Here, the number of workers is greater than tasks' demand.

are raised. In addition, TF-N, TF-LO, and WF-LO do not consider the whole candidate set and some candidates might nearly never be chosen.

When the task number increases and candidates will be in shortage, as shown in Figure 7, every worker has more options of choosing tasks and the satisfaction degree of using WF-GO is still the highest. When the task number increases, candidates easily obtain plenty of light tasks, so the satisfaction degrees of all methods increase. In the case of a lack of candidates, TF-N and TF-LO have a similar result of the satisfaction degree.

As the satisfaction degree is related to the expected task number, we adjust this number and show the experimental result in Figure 8. On one hand, with enough candidates, the allocation ratios of all methods are nearly 100%. On the other hand, if all candidates expect more tasks, the mean satisfaction degree will decrease. As such, the advantage of using WF-GO becomes greater than the others. To conclude, we

recruit less workers through using WF-GO which will save the labor cost. In this way, workers will get their expected number of tasks leading to satisfied experience with task allocation.

## Conclusion

Task allocation is an important step for the MCS applications with a limited labor budget. Constrained by the spatial and temporal situation of both candidates and tasks, a task allocation method must consider the spatial and temporal relationship between candidates and tasks to maximize the number of allocated tasks as well as minimizing the gap between the number of assigned tasks and the number of required tasks for every worker (i.e. satisfying workers). We define two optimization objectives for this task allocation problem and develop two greedy-based task allocation algorithms to both worker-first and task-first methods. Experiments are conducted and results show that global-optimizing algorithms increase both the task allocation ratio and the worker satisfaction degree. How to minimize workers' detour distances and lower the total payment will be our future research effort.


## Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially supported by the National Natural Science Foundation of China (No. 61972092, No. 61602230, No. 61772428, and No. 61725205).

## ORCID iD

Huihui Chen  <https://orcid.org/0000-0003-4465-2716>

## References

- Guo B, Wang Z, Yu Z, et al. Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm. *ACM Comput Sur* 2015; 48(1): 7.
- Chen H, Guo B, Yu Z, et al. A generic framework for constraint-driven data selection in mobile crowd photographing. *IEEE Internet Things J* 2017; 4(1): 284–296.
- Guo B, Chen H, Yu Z, et al. FlierMeet: a mobile crowd-sensing system for cross-space public information reposting, tagging, and sharing. *IEEE Trans Mob Comput* 2015; 14(10): 2020–2033.
- Chen H, Guo B, Yu Z, et al. Mobile crowd photographing: another way to watch our world. *Sci China Inform Sci* 2016; 59(8): 083101.
- Guo B, Han Q, Chen H, et al. The emergence of visual crowdsensing: challenges and opportunities. *IEEE Commun Surv Tut* 2017; 19(4): 2526–2543.
- Zhao Y, Li Y, Wang Y, et al. Destination-aware task assignment in spatial crowd-sourcing. In: *Proceedings of the 2017 ACM on conference on information and knowledge management (CIKM)*, Singapore, 6–10 November 2017, pp.297–306. New York: ACM.
- Chen H, Cao Y, Guo B, et al. LuckyPhoto: multi-facet photographing with mobile crowdsensing. In: *Proceedings of the 2018 ACM international joint conference and 2018 international symposium on pervasive and ubiquitous computing and wearable computers*, Singapore, 8–12 October 2018, pp.29–32. New York: ACM.
- Chen H, Guo B, Yu Z, et al. Crowdtracking: real-time vehicle tracking through mobile crowdsensing. *IEEE Internet Things* 2019; 6(5): 7570–7583.
- Zhang D, Wang L, Xiong H, et al. 4W1H in mobile crowd sensing. *IEEE Commun Mag* 2014; 52(8): 42–48.
- Kandappu T, Jaiman N, Tandriansyah R, et al. Tasker: behavioral insights via campus-based experimental mobile crowd-sourcing. In: *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing (Ubicomp)*, Heidelberg, 12–16 September 2016, pp.392–402. New York: ACM.
- Chen C, Zhang D, Ma X, et al. Crowddeliver: planning city-wide package delivery paths leveraging the crowd of taxis. *IEEE Trans Intell Transp Syst* 2017; 18(6): 1478–1496.
- Wang J, Wang L, Wang Y, et al. Task allocation in mobile crowd sensing: state of the art and future opportunities. *IEEE Internet Things J* 2018; 5(5): 3747–3757.
- Wang L, Yu Z, Guo B, et al. Mobile crowd sensing task optimal allocation: a mobility pattern matching perspective. *Front Comput Sci* 2018; 12(2): 231–244.
- Ji S, Zheng Y and Li T. Urban sensing based on human mobility. In: *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing (Ubicomp)*, Heidelberg, 12–16 September 2016, pp.1040–1051. New York: ACM.
- Liu Y, Guo B, Wang Y, et al. TaskMe: multi-task allocation in mobile crowd sensing. In: *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing (Ubicomp)*, Heidelberg, 12–16 September 2016, pp.403–414. New York: ACM.
- Yang D, Zhang D and Qu B. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Trans Intell Syst Technol* 2016; 7(3): 30.
- Guo B, Liu Y, Wu W, et al. Active-crowd: a framework for optimized multitask allocation in mobile crowdsensing systems. *IEEE Trans Hum Mach Syst* 2017; 47(3): 392–403.
- Zhang D, Xiong H, Wang L, et al. CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint. In: *Proceedings of 2014 ACM international joint conference on pervasive and ubiquitous computing (Ubicomp)*, Seattle, WA, 13–17 September 2014, pp.703–714. New York: ACM.

19. Wang L, Zhang D, Wang Y, et al. Sparse mobile crowdsensing: challenges and opportunities. *IEEE Commun Mag* 2016; 54(7): 161–167.
20. Cheng P, Lian X, Chen Z, et al. Reliable diversity-based spatial crowdsourcing by moving workers. *Proceedings VLDB Endowment* 2015; 8(10): 1022–1033.
21. Chen C, Jiao S, Zhang S, et al. TripImputor: real-time imputing taxi trip purpose leveraging multi-sourced urban data. *IEEE Trans Intell Transp Syst* 2018; 19: 3292–3304.
22. Xiong H, Zhang D, Chen G, et al. iCrowd: near-optimal task allocation for piggyback crowdsensing. *IEEE Trans Mob Comput* 2016; 15(8): 2010–2022.
23. Reddy S, Estrin D and Srivastava M. Recruitment framework for participatory sensing data collections. In: *Proceedings of international conference on pervasive computing (Percom)*, Helsinki, 17–20 May 2010, pp.138–155. Berlin: Springer.
24. Xiao M, Wu J, Huang L, et al. Multi-task assignment for crowdsensing in mobile social networks. In: *Proceedings of the 2015 IEEE conference on computer communications (INFOCOM)*, Kowloon, Hong Kong, 26 April–1 May 2015, pp.2227–2235. New York: IEEE.
25. Karaliopoulos M, Telelis O and Koutsopoulos I. User recruitment for mobile crowdsensing over opportunistic networks. In: *Proceedings of the 2015 IEEE conference on computer communications (INFOCOM)*, Kowloon, Hong Kong, 26 April–1 May 2015, pp.2254–2262. New York: IEEE.
26. Musthag M and Ganesan D. Labor dynamics in a mobile micro-task market. In: *Proceedings of the SIGCHI conference on human factors in computing systems*, Paris, 27 April–2 May 2013, pp.641–650. New York: ACM.
27. Wang Z, Tan R, Hu J, et al. Heterogeneous incentive mechanism for time-sensitive and location-dependent crowdsensing networks with random arrivals. *Comput Netw* 2018; 131: 96–109.
28. Wang J, Wang F, Wang Y, et al. Allocating heterogeneous tasks in participatory sensing with diverse participant-side factors. *IEEE Trans Mob Comput* 2019; 18: 1979–1991.