Received: 00 Month 0000 | Revised: 00 Month 0000 | Accepted: 00 Month 0000

DOI: xxx/xxxx

RESEARCH ARTICLE

Task Bundling in Worker-Centric Mobile Crowdsensing

Tianlu Zhao[1] | Yongjian Yang[1] | En Wang*[1] | Shahid Mumtaz[2] | Xiaochun Cheng[3]

[1]Department of Computer Science and Technology, Jilin University, Changchun, China
[2]Institute of Telecommunications, DETI, University of Aveiro, Aveiro, Portugal
[3]Department of Computer Communications, Middlesex, University, London, U.K.

Correspondence
*En Wang, Department of Computer Science and Technology, Jilin University, Changchun, Jilin 130012, China. Email: wangen@jlu.edu.cn

Abstract

Most existing researches about task allocation in mobile crowdsensing mainly focus on requester-centric mobile crowdsensing (RCMCS), where the requester assigns tasks to workers to maximize his/her benefits. A worker in RCMCS might suffer benefit damage because the tasks assigned to him/her may not maximize his/her benefit. Contrarily, worker-centric mobile crowdsensing (WCMCS), where workers autonomously select tasks to accomplish to maximize their benefits, does not receive enough attention. The workers in WCMCS can maximize their benefits, but the requester in WCMCS will suffer benefit damage (cannot maximize the number of expected completed tasks). It is hard to maximize the number of expected completed tasks in WCMCS, because some tasks may be selected by no workers, while others may be selected by many workers. In this paper, we apply task bundling to address this issue, and we formulate a novel task bundling problem in WCMCS with the objective of maximizing the number of expected completed tasks. To solve this problem, we design an algorithm named LocTrajBundling which bundles tasks based on the location of tasks and the trajectories of workers. Experimental results show that, compared with other algorithms, our algorithm can achieve a better performance in maximizing the number of expected completed tasks.

KEYWORDS:
mobile crowdsensing, task bundling, NP-Hard, simulated annealing, heuristic algorithm

1 | INTRODUCTION

In recent years, the proliferation of sensor-rich mobile devices (e.g., smart phones) has stimulated the development of mobile crowdsensing (MCS)[1 2 3 4], an effective and efficient sensing paradigm which exploits a large number of workers with intelligent mobile devices to perform numerous sensing tasks in real world. Compared with traditional urban sensing paradigm, mobile crowdsensing avoids the deployment of expensive static infrastructure (e.g., traffic monitoring system, air condition monitoring stations); it utilizes the mobility of workers and the sensors (e.g., GPS, camera) embedded in intelligent mobile devices to achieve higher flexibility, lower cost and wider sensing coverage. The advantages of mobile crowdsensing make it widely used in various fields such as air quality monitoring[5 6], target tracking[7], traffic monitoring[8 9], digital map updating[10], and many crowdsensing systems have been constructed and applied in real life[11 12 13 14 15 16].

There are two basic components in a typical mobile crowdsensing system: one or more requesters and a number of workers. Requesters have some tasks to be completed, and they will publish the tasks to numerous workers; workers get rewards by completing the tasks published by requesters and uploading the data they collect. For requesters, they want as many tasks to be completed as possible, while for workers, they want to maximize their benefits at acceptable costs. As an example, in the

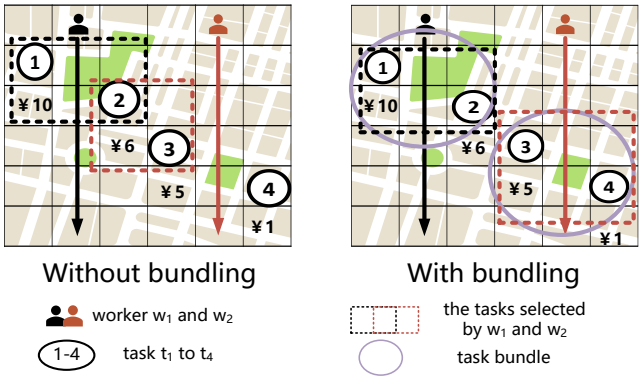


FIGURE 1 Task bundling is conducive to increase the number of expected completed tasks. There are two workers w_1, w_2, and four tasks t_1, t_2, t_3, t_4 with rewards 10, 6, 5, 1, respectively. Suppose that w_1 is able to complete t_1, t_2 and w_2 is able to complete t_2, t_3 or t_3, t_4 due to some limitations. In WCMCS, without bundling, only three tasks of the four tasks will be completed due to the individual rationality. Specifically, to maximize their profits, w_1 will select task t_1 and t_2, and w_2 will select task t_2 and t_3, while t_4 will not be completed. After we bundle t_1 and t_2 as bundle 1 and bundle t_3 and t_4 as bundle 2, w_1 will select bundle 1 to complete t_1 and t_2, w_2 will select bundle 2 to complete t_3 and t_4, so that all the four tasks will be completed.

scenario where workers have their own trajectories and tasks are location-dependent, in order to complete a task, a worker needs to detour to the location of this task, which will cause detour cost. As a result, in situations where tasks are nearly homogeneous, a worker is more willing to perform tasks nearby his/her trajectory.

A key issue in mobile crowdsensing is task assignment. According to the different ways of task assignment, mobile crowdsensing can be divided into two categories: requester-centric mobile crowdsensing (RCMCS) and worker-centric mobile crowdsensing (WCMCS). In RCMCS, the requester formulates a task assignment plan to determine what tasks each worker needs to accomplish, and he/she assigns tasks to each worker based on the task assignment plan made by itself. On the contrary, in WCMCS, each worker autonomously decides what tasks he/she will accomplish, and the requester does not have the authority to formulate and interfere with each worker's task assignment plan. In other words, each worker assigns tasks to itself.

There have been several studies that focus on the task assignment in mobile crowdsensing [17 18 19 20 21]. However, most studies in terms of task assignment in mobile crowdsensing mainly concentrate on the RCMCS scenario, in which the requesters' benefit can be maximized, but the workers might suffer benefit damage. For instance, in Fig. 1 , there is a requester Q, two workers w_1, w_2, and four tasks t_1, t_2, t_3, t_4 with rewards 10, 6, 5, 1, respectively. w_1 is able to complete t_1, t_2, and w_2 is able to complete t_2, t_3 or t_3, t_4. In RCMCS, the requester Q will assign t_1, t_2 to w_1, and assign t_3, t_4 to w_2 so that all the tasks can be completed under the worker ability constraint. This task execution plan is optimal for the requester, but not for the workers. It can be seen that, for w_2, completing task t_2 and t_3 can get a total reward of 11, which can maximize his/her profit; however, he/she was finally assigned task t_3 and t_4 to get a total reward of 6, which results in the loss of w_2's benefit.

The limitation of traditional RCMCS makes it need to be complemented by WCMCS, which draws little attention nowadays [22]. Compared with RCMCS, it can be seen that workers enjoy full autonomy in making task execution plan in WCMCS. However, due to the individual rationality of workers, each worker is more likely to select the tasks that are nearer to his/her trajectory or with higher rewards. It will lead to the fact that some tasks may be selected by numerous workers while some tasks may be selected by no workers and the requesters will suffer benefit damage, so that it is hard to maximize the number of expected completed tasks in WCMCS.

In this paper, we focus on the problem of maximizing the number of expected completed tasks in worker-centric mobile crowdsensing to maximize the benefit of requesters, and we apply task bundling mechanism to solve this problem. The basic idea of task bundling is to bundle all the independent tasks as several task bundles according to some attributes, such as the location of tasks, the trajectory of each worker, and the reward of each task; each task bundle consists of several independent tasks. For a task bundle, each worker has two choices: to complete all the tasks in the bundle and get rewards, or not to complete any tasks in the bundle. We argue that task bundling is a helpful way to maximize the number of expected completed tasks as possible. The reason for this phenomenon is twofold. First, some unpopular tasks (selected by few workers) are more likely to be completed if they are bundled with popular tasks (selected by many workers), because workers cannot just complete the popular tasks in a task bundle. Second, the reward of a task bundle is higher than any independent task in this task. Therefore, compared

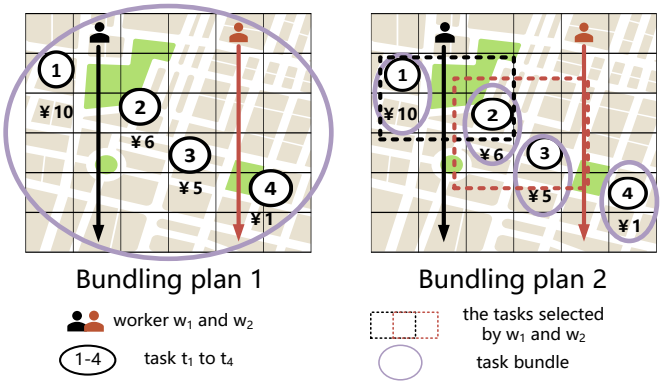


FIGURE 2 It is challenging to balance the trade-off between the number of bundles and the size of each bundle to find the optimal task bundling plan. If we bundle all the four tasks as a bundle, there will be only one bundle, and no tasks will be completed. If we bundle each task as a bundle with only one task, there will be four task bundles, and only three tasks will be completed. Neither of these two task bundling plans is the optimal task bundling plan.

with completing an independent task, workers are more willing to complete a task bundle and get higher rewards. Third, in some scenario where workers have some specific kinds of cost, we can use task bundling to induce workers to complete unpopular tasks instead of popular tasks.

For example, in Fig. 1 , w_1 is able to complete t_1, t_2, and w_2 is able to complete t_2, t_3 or t_3, t_4 due to some limitations. On condition that the scenario is WCMCS, w_1 will select task t_1 and t_2, and w_2 will select task t_2 and t_3; thus, t_4 will not be selected by any worker. With task bundling, we can bundle t_1 and t_2 as bundle 1 and bundle t_3 and t_4 as bundle 2. In that way, w_1 will select bundle 1 to complete t_1 and t_2; w_2 will select bundle 2 to complete t_3 and t_4.

It is quite challenging to find the optimal task bundling plan which can maximize the number of expected completed tasks in WCMCS due to the following reasons. First, to some extent, moderate task bundling helps to improve the number of expected completed tasks. However, a task bundling plan which includes fewer and bigger task bundles does not mean a better plan, nor does the task bundling plan which includes more and smaller task bundles. For example, in Fig. 2 , if we bundle all the four tasks as a bundle, no tasks will be completed. If we bundle each task as a bundle with only one task, only three tasks will be completed. From Fig. 1 , we know that the best task bundling plan is to bundle t_1 and t_2 as bundle 1 and bundle t_3 and t_4 as bundle 2; neither of the two task bundling plans in Fig. 2 is the optimal task bundling plan. Thus, how to *balance the trade-off between the number of bundles and the size of each bundle* is the first challenge. Second, when making the task packing plan, we should not only consider the relevant attributes of the tasks, but also consider the workers' trajectories. This makes it more difficult for us to find the optimal task bundling plan, which is the second challenge.

The motivation for us to write this article is to find an optimal task bundling plan which can maximize the number of expected completed tasks in WCMCS to maximize the benefit of requesters, while few researches take note of this kind of problem nowadays. We mathematically formulate a task bundling problem which focuses on maximizing the number of expected completed tasks in WCMCS. Then we prove that this problem is NP-Hard. To solve this problem, we propose a heuristic algorithm named LocTrajBundling which includes two phases: initial solution construction phase that constructs an initial solution by greedy algorithm, simulated annealing optimization phase that optimizes the initial solution by running simulated annealing algorithm iteratively. We obtain a real-world dataset which is used for our real-world dataset experiments by using BaiduMap API, and this dataset includes the location of 400 hotels and the minimum walking distance between each pair of hotels. The experimental results reveal that our algorithm has a better performance than the other contrast algorithms in maximizing the number of expected completed tasks in WCMCS. Briefly speaking, we have made the following contributions:

- We argue that task bundling is a feasible way to find an optimal task bundling plan in worker-centric mobile crowdsensing. Then we formulate a novel task bundling problem in WCMCS which aims to maximize the number of expected completed tasks. As far as we know, it is the first work that focuses on maximizing the number of expected completed tasks in worker-centric mobile crowdsensing.

- We prove the NP-Hardness of this problem and propose a heuristic algorithm named LocTrajBundling to solve this problem. This algorithm consists of two phases: initial solution construction phase and simulated annealing optimization

phase. The first phase uses a greedy algorithm to generate an initial solution, and the second phase runs simulated annealing algorithm to optimize the initial solution. In order to optimize the solution obtained by the second phase, this algorithm will repeat the second phase for many times until the increase of the number of expected completed tasks approaches zero.

- A real-world dataset obtained from BaiduMap API and a synthetic dataset are tested in our experiments, and the experimental results show that, in both the real-world dataset experiments and the synthetic dataset experiments, the task bundling plan provided by our algorithm can achieve a better performance in maximizing the number of expected completed tasks than other contrast algorithms.

The rest of the paper is organized as follows. We introduce the related work in Section II. Some basic preliminaries are listed in Section III and the worker behavior model is presented in Section IV. In Section V, we mathematically formulate the task bundling problem that focuses on maximizing the number of expected completed tasks in WCMCS. Then, in Section VI, we provide our heuristic algorithm named LocTrajBundling for solving the task bundling problem. We do several experiments to evaluate the performance of our algorithm and show the results in Section VII. Finally, we conclude this paper in Section VIII.

2 | RELATED WORK

2.1 | Task assignment in mobile crowdsensing

Task assignment is a crucial problem in mobile crowdsensing and has been studied for many years[23 24 25 26 27]. So far, most researches mainly have focused on the task assignment problem in requester-centric mobile crowdsensing, in which requesters determine what tasks each worker must complete, and workers passively complete the tasks assigned by requesters. Essentially speaking, task assignment problems in MCS are almost the mathematical optimization problems with various goals and constraints, such as sensing quality[28 29 30 31], incentive cost[32 33], location privacy[34 35 36 37 38 39], and social surplus[40]. For example, [32] focused on the problem in task assignment which aimed to minimize the incentive cost under the minimum level of sensing quality constraint. In terms of different classification standards, the problem of task assignment in MCS can have different classification[41]. For instance, according to the number of objectives in the problem, we can classify the task assignment problem as single-objective-oriented[42 32] and multi-objective-oriented[43 44]. We can also classify the task assignment problem as offline[42 40] and online[45] based on the time when the task assignment plan is formulated. However, as we mentioned above, most current works about task assignment in MCS mainly follow with interest the RCMCS scenario, and few researches pay attention to task assignment problem in worker-centric mobile crowdsensing.

2.2 | Task bundling in mobile crowdsourcing

Task bundling aims to bundle independent tasks as several task bundles, so that a worker who selects a bundle has to complete all the tasks in this bundle to get reward, otherwise he/she will get no reward. Some researches have applied task bundling into crowdsourcing[46], which is a research field that is highly relevant to mobile crowdsensing. In[47], the authors aimed to bundle tasks according to the context of each worker, and then recommended task bundles to workers based on their context to maximize the number of completed tasks and reduce task costs. However, the problem in[47] is different from ours, because the workers in[47] can only make decisions on the task bundles recommended to him/her, in other words, the workers cannot select and complete the bundles that are not recommended to him/her. Beyond that, the workers can get paid even though they have completed only a portion of the tasks in a task bundle, which is also different from our work. It is shown in[48] that workers prefer bundled tasks even though the reward of a task bundle is smaller than the sum of the reward of each independent task, and the average completion rate of tasks has increased by 19% after task bundling. But the task bundling algorithm proposed by[48] just bundles the tasks on the same floor of a building, which is not suitable for the problem in our paper. In[49], the authors presented a task bundling incentive mechanism to address the participation unbalance problem in location dependent mobile crowdsourcing, and the reward of a bundle is equal to the sum of the reward of each independent task. However, the algorithm in[49] artificially categorizes each task into a cold task or a hot task and ensures that each task bundle contains both hot tasks and cold tasks. In other words, each cold task must be bundled with a hot task, thus a task bundle containing only cold tasks is not allowed. By contrast, in this paper, although the probability of each task being completed is different, we do not classify each task into hot task or cold task according to the probability of being completed, and any different tasks can be bundled into a bundle. In addition, the algorithm in[49] runs without knowing the location and trajectory of each worker, which is also different from our

work. To summarize, all the aforementioned works are different from our work even if they concentrate on task bundling like our work, which motivates us to write this paper.

3 | PRELIMINARIES

In our problem, there are m workers denoted as $W = \{w_1, w_2, \dots, w_m\}$; each worker has a starting point s, a destination d, a travel distance budget bud, and a trajectory $traj$ from its starting point to its destination. The trajectory of a worker represents the shortest path from its starting point to its destination. The starting point set is $S = \{s_1, s_2, \dots, s_m\}$. The destination set is $D = \{d_1, d_2, \dots, d_m\}$. The travel distance budget set is $Bud = \{bud_1, bud_2, \dots, bud_m\}$, and the trajectory set is $Traj = \{traj_1, traj_2, \dots, traj_m\}$ (the definition of trajectory will be given in the following graphs). The requester has n tasks of the same type (e.g., air pollution collection), and the n tasks are denoted as $T = \{t_1, t_2, \dots, t_n\}$; each task has two properties: l represents the location of the task, and r represents the reward of this task. The set which includes the location of each task is $L = \{l_1, l_2, \dots, l_n\}$, and the reward set is $R = \{r_1, r_2, \dots, r_n\}$. The reward of an arbitrary task t is $r(t)$.

The requester knows the information of all the workers and will make a task bundling plan according the information to maximize the number of expected completed tasks. Each worker only knows the information of itself, and he/she will select tasks to complete after the task bundling plan is made by the requester. To complete a task, a worker need to arrive at the location of the task, collect data and upload the data to the requester. A task t can be completed by many workers, but each worker can complete the task t only once. In order to improve the enthusiasm of workers and ensure the accuracy of data, we allow for data redundancy, and everyone who completes the task t will get the same reward $r(t)$.

Both S, D and L are parts of a traffic network, which is a complete directed graph $G = (V, E)$. V is the vertex set. A vertex v in V represents a pair of coordinates (x, y) to show its spatial location, where x is the longitude coordinate of v and y is the latitude coordinate of v. E is the edge set including $|V|^2$ elements since G is a complete directed graph. Each element e in E represents the road length (defined below) between two specific vertices and the cost of a element e is $c(e)$.

*Definition 1 (Road Length).*In a traffic network $G = (V, E)$, the road length between vertices i and j in set V, denoted as rl_{ij}, is the minimal walking distance between the two vertices. In the real world, rl_{ij} is not equal to the linear distance between i and j in most cases. In this paper, we calculate the minimal walking distance by aid of BaiduMap API. We take the latitude and longitude coordinates of i and j as input, and get the calculation result through the calculation of BaiduMap API, denoted as rl_{ij}, which represents the road length between vertex i and vertex j.

*Definition 2 (Trajectory).*The trajectory of worker w_i, denoted as $traj_i =< traj_{i1}, traj_{i2}, \dots, traj_{in} >$, is a sequence of vertices, and each vertex is an element in V. Since $G = (V, E)$ is a complete directed graph, any two consecutive vertices v_1, v_2 are directly connected by an edge $e(v_1, v_2)$ that represents the road length between the two vertices. At first, each trajectory only includes two vertices which represent the starting point and the destination of worker w_i.

*Definition 3 (Trajectory cost).*Given a trajectory $traj_i =< traj_{i1}, traj_{i2}, \dots, traj_{in} >$, the cost of the trajectory is the sum of the costs of the edges in this trajectory, i.e.,

$$TC(traj_i) = \sum_{j=1}^{n-1} c(e(traj_{ij}, traj_{i(j+1)})). \tag{1}$$

It is obvious that $TC(traj_i) \leq bud_i$.

*Definition 4 (Task Bundling Plan).*A task bundling plan is a set $B = \{b_1, b_2, \dots, b_k\}$ which is used to determine which tasks are bundled in the same bundle; each element in the set B represents a task bundle that consists of at least one task. $|b_i|$ represents the number of tasks in the bundle b_i.

Definition 5 (Reward of a bundle). The reward of a bundle b is the sum of the rewards of the tasks in this bundle, i.e.,

$$RB(b) = \sum_{t \in b} r(t). \tag{2}$$

4 | WORKER BEHAVIOR MODEL

A worker can autonomously select a task bundle and complete the tasks in the bundle to get rewards, and he/she has to detour from his/her original trajectory to complete the tasks in the bundle without changing his/her starting point and destination. We use $traj_i \bigoplus b_k$ to represent the trajectory of a worker w_i after he/she choose bundle b_k. We simply use a greedy-based method to determine the sequence of the vertices in the newly-generated trajectory: First, we add the starting point into the newly-generated trajectory, and then we find the task t' whose location is nearest to the starting point and add the location vertex v' into the trajectory, next we find the task t'' whose location is nearest to v' and add the location vertex v'' into the trajectory, and so on. Finally, we add the destination into the trajectory.

We assume that no task is on the trajectory of any worker. In this case, a worker must detour to complete a task, which will cause detour cost. As noted earlier, each worker is more likely to select the tasks that are nearer to his/her trajectory or with higher rewards due to the individual rationality of workers. To calculate the probability of a worker w_i selecting a task bundle b_k, we give the following worker behavior model that formulates the probability:

$$P(w_i, b_k) = \begin{cases} \dfrac{TC(traj_i)}{TC(traj_i \bigoplus b_k)} * \dfrac{RB(b_k)}{\sum_{j=1}^{n} r_j} * \dfrac{\sum_{t \in b_k} TC(traj_i \bigoplus t)}{TC(traj_i \bigoplus b_k) * |b_k|}, & TC(traj_i \bigoplus b_k) \le bud_i, \\ 0, & TC(traj_i \bigoplus b_k) > bud_i. \end{cases} \tag{3}$$

It can be seen that this formula includes three parts when $TC(traj_i \bigoplus b_k) \le bud_i$. The first part shows that the smaller the detour cost, the greater the probability of w_i completing the task bundle b_k. The second part shows that workers are more willing to select a task bundle with higher rewards. The third part shows that workers are more likely to select a task bundle whose tasks are more centrally distributed. In particular, we have to explain the third part of this formula in detail. $TC(traj_i \bigoplus t)$ represents the cost for worker w_i to complete one specific task t in this bundle. There are $|b_k|$ tasks in this bundle, and the numerator of the third part represents the sum of the $|b_k|$ costs. The smaller value of the third part, the more sparsely distributed of the tasks in this bundle. If the location of all the tasks are the same, then the value of the third part is 1, which is the maximum of the third part.

The probability of a task bundle b_k to be completed is actually the probability of a task bundle to be completed by at least one worker, so the probability can be formulated as follows:

$$P(b_k) = 1 - \prod_{i=1}^{n} (1 - P(w_i, b_k)). \tag{4}$$

Consequently, the number of expected completed tasks of a task bundle is $|b_k| * P(b_k)$.

5 | PROBLEM FORMULATION

Based on the above preliminaries and worker behavior model, we can formulate the task bundling problem as follows:

In a traffic network which is a complete directed graph $G = (V, E)$, given the locations and rewards of n tasks $T = \{t_1, t_2, \dots, t_n\}$, the trajectories of m workers $W = \{w_1, w_2, \dots, w_m\}$, and the travel distance budget set $Bud = \{bud_1, bud_2, \dots, bud_m\}$, the requester aims to find a task bundling plan B that can maximize the number of expected completed tasks:

$$\text{Maximize} \sum_{b \in B} (|b| * P(b)) \tag{5}$$

$$\text{Subject to}: \bigcup_{b \in B} b = T, \tag{6}$$

$$\bigcap_{b \in B} b = \emptyset, \tag{7}$$

$$b \neq \emptyset \qquad \forall b \in B. \tag{8}$$

Theorem 1. The task bundling problem is NP-Hard.

Apparently, if a special case of the task bundling problem is NP-hard, then the task bundling problem is NP-hard. We consider a special case of the task bundling problem where all the tasks can only be bundled as at most two bundles, and we call this problem "special task bundling problem". Next, we formulate another problem named "binary task bundling problem" which is

equivalent to the special task bundling problem. As we mentioned before, in binary task bundling problem, all the tasks can only be bundled as at most two bundles, and we can use a n-dimensional vector $< x_1, x_2, \dots, x_n >$ to represent different task bundling plan. The value of each x can only be 0 or 1. For example, $< 1, 1, ..., 1 >$ and $< 0, 0, ..., 0 >$ represent that all the tasks are bundled as only one bundle, and there is no tasks in the other bundle. So the binary task bundling problem is defined as follows:

Find a feasible solution $< x_1, x_2, \dots, x_n >$ that:

$$\text{Maximize} \sum_{k=1}^{2}(|b_k| * P(b_k)) \tag{9}$$

$$\text{Subject to}: \quad x_i \in \{0, 1\} \qquad \forall i \in \{1, 2, \dots, n\}. \tag{10}$$

The binary task bundling problem is equivalent to the special task bundling problem, and each solution of the binary task bundling problem can be transformed to an equivalent solution of the special task bundling problem. The value of each variable can only be 0 or 1, so the binary task bundling problem is an integer programming problem. The objective function can be regarded as a function $f(x_1, x_2, \dots, x_n)$ with n inputs. According to the above formulas, we can see that this function $f(x_1, x_2, \dots, x_n)$ is a nonlinear function. Therefore, it can be concluded that the binary task bundling problem is a nonlinear integer programming problem, which is proved to be a NP-Hard problem[50]. The special task bundling problem is equivalent to the binary task bundling problem, so the special task bundling problem is NP-Hard, thus the task bundling problem is NP-Hard ■.

Since the task bundling problem is NP-hard, the number of all possible bundling solutions increases sharply with the number of tasks and workers increase. Actually, finding a task bundling plan is essentially dividing the task set into several non-empty subsets so that every task is included in only one subset, and a task bundling plan is essentially "a partition of a set". It is proved that a set with n elements has B_n partitions, where B_n is the bell number[51]. The calculation of bell number is

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_n. \tag{11}$$

The first eleven bell numbers are 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975. The fast growth of bell numbers and the NP-Hardness of the task bundling problem make it impossible to find an optimal solution in polynomial time. Accordingly, a heuristic algorithm is presented in this paper for solving the task bundling problem.

6 | LOCTRAJBUNDLING: A HEURISTIC ALGORITHM

In this section, we present a heuristic algorithm named LocTrajBundling which includes two phases: initial solution construction phase and simulated annealing optimization phase. The initial solution generated in the first phase will be used as the input of the second phase. First, we give a brief description and the pseudo-code of the two phases of our heuristic algorithm, then we give the pseudo-code of LocTrajBundling.

6.1 | Initial solution construction

It is hard to find a good initial solution because we need to face the following challenges: 1. how to determine the size of each task bundle? If the size of a task bundle is too small, it will not be able to provide enough incentives for workers; however, if a task bundle is too large, it will also reduce the willingness of workers to choose this task bundle. 2. how to make a task bundle plan according to the trajectories of workers? After all, tasks will be completed by the workers. When formulating the task bundling plan, we should not only consider the location of the tasks, but also consider the workers' trajectories. Taking the aforementioned challenges into account, we firstly construct a relatively good initial solution by a greedy-based algorithm to provide an input for the simulated annealing optimization phase.

We firstly give the definition of ALOW probability before we introduce the initial solution construction algorithm.

Definition 6 (ALOW probability) The ALOW(at least one worker) probability of a task t is the probability that the task can be completed by at least one worker.

It can be seen that, when we calculate the ALOW probability of a task, we need to use the location of this task, the reward of this task, and the trajectory of all workers. As we mentioned before, the probability of a task bundle b_k to be completed by at least one worker is:

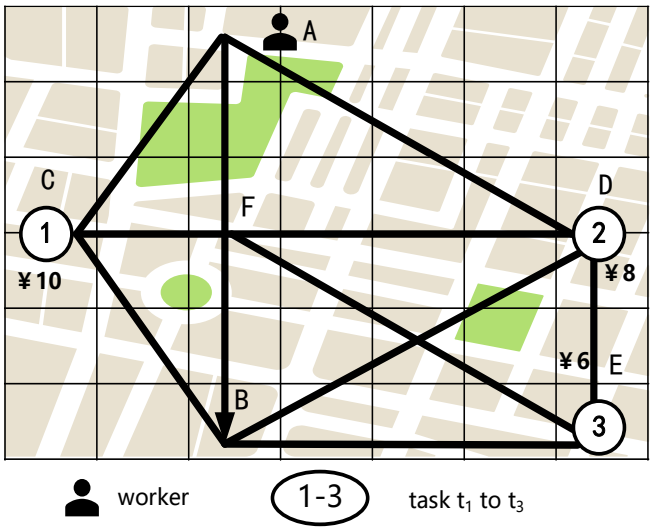


FIGURE 3 There is one worker w_1, and three tasks t_1, t_2, t_3 with rewards 10, 8, 6, respectively. The worker's travel distance budget is 100. A and B are the starting point and the destination of the worker. The coordinates of A are (2.25, 6), and the coordinates of B are (2.25, 0). The location of the three tasks t_1, t_2, t_3 are C, D, E. The coordinates of C, D, E are (0, 3), (6.25, 3), (6.25, 0). Line AB intersects line CD at F, whose coordinates are (2.25, 3).

$$P(b_k) = 1 - \prod_{i=1}^{n}(1 - P(w_i, b_k)).$$

We can regard a single task t as a task bundle with only one task, so the ALOW probability of a task t is as follows:

$$ALOW_t = P(\{t\}) = 1 - \prod_{i=1}^{n}(1 - P(w_i, \{t\})). \tag{12}$$

Now we will introduce the initial solution construction algorithm, the main idea of the algorithm is as follows:

First, we sort the task set T to get a queue Q according to the value of ALOW probability from large to small.

Second, we get the first element of the queue (denoted as q') and add q' to an empty set S_1, then we remove q', the first element of the queue. If the queue is empty after removing the first element of the queue, then we add S_1 into the task bundling set TBP, and the algorithm terminates; if not, the algorithm will go to the third step.

Third, we find out the element q'' that can maximize the increase of the number of expected completed tasks if we bundle q'' with S_1. If the maximum value of the increase is greater or equal to 0, then we add the q'' into S_1, and remove q'' from the queue Q, and if Q is empty after removing q'', then we add S_1 into the task bundling plan set TBP, and the algorithm terminates. Conversely, if the maximum value of the increase is less than 0, we clone S_1 into S_1' and add S_1' into the task bundling plan set TBP, then we clear S_1 to an empty set. We repeat the second step and the third step until there is no element in the queue. This algorithm comprehensively considers the workers' trajectories and location of tasks to construct an initial task bundling plan due to the usage of ALOW probability, and it reasonably determines the size of each task bundle.

We use Fig. 3 to describe the initial solution construction algorithm more graphically. In Fig. 3 , if two points are connected by an edge, it means that a worker can directly reach another point from one point. There is one worker with starting point A and destination B. The worker's travel distance budget is 100 so that the trajectory cost will not exceed the travel distance budget even if the worker selects all the three tasks. The coordinates of A are (2.25, 6), and the coordinates of B are (2.25, 0). There are three tasks t_1, t_2, t_3 with location C, D, E. The coordinates of C, D, E are (0, 3), (6.25, 3), (6.25, 0), and the rewards of t_1, t_2, t_3 are 10, 8, 6. Line AB intersects line CD at F, the coordinates of F are (2.25, 3).The task bundling plan set is empty at first. We firstly sort the three tasks according to the value of ALOW probability from large to small. The ALOW probabilities of t_1, t_2, t_3 are $\frac{1}{3}$, $\frac{1}{5}$, $\frac{1}{8}$, respectively. Therefore, the task sequence Q after sorting is t_1, t_2, t_3, so the task queue is $< t_1, t_2, t_3 >$. Without bundling, the number of expected completed tasks is $\frac{1}{3} * 1 + \frac{1}{5} * 1 + \frac{1}{8} * 1 = \frac{79}{120}$. Now we get t_1 which is the first element of Q, then we add t_1 to an empty set S_1 and remove t_1 from Q. Now Q is $< t_2, t_3 >$, and we have to find an element that can maximize

the increase of the number of expected completed tasks from Q. If we choose t_2, the value of the increase is ($\frac{6}{15}$*$\frac{18}{24}$*$\frac{7}{12}$ * 2) - ($\frac{1}{3}$ + $\frac{1}{5}$) = -$\frac{11}{60}$; if we choose t_3, the value of the increase is ($\frac{6}{15}$ * $\frac{16}{24}$ * $\frac{13}{24}$ * 2) - ($\frac{1}{3}$ + $\frac{1}{8}$) = -$\frac{67}{600}$. So the maximum value of the increase is -$\frac{67}{600}$ <0, this means that neither task t_2 nor task t_3 is suitable to be bundled with task t_1, so we clone S_1 into S_1' and add S_1' into the task bundling plan set, then we clear S_1 to an empty set. Now the task queue Q is $< t_2, t_3 >$, the task bundling plan set TBP is t_1.

Next, we repeat the second step and the third step of this algorithm. We get t_2 from Q and add t_2 to S_1, then we remove t_2 from Q. Now there is only one element t_3 in Q. We choose t_3 to calculate the value of the increase of the number of expected completed tasks after bundling, which is ($\frac{6}{12}$ * $\frac{14}{24}$ * $\frac{11}{12}$ * 2)-($\frac{1}{5}$ + $\frac{1}{8}$)= $\frac{151}{720}$ >0. Therefore, we add t_3 to S_1 and remove t_3 from Q. Q is empty after removing t_3 from itself, so we add S_1 into the task bundling set TBP, and the algorithm terminates. Finally, the task queue Q becomes empty, and we get a task bundling plan set $TBP = \{\{t_1\}, \{t_2, t_3\}\}$. That is to say, we will bundle task t_2 and task t_3 as a task bundle, t_1 is regarded as a task bundle in which there is only one tasks. The number of expected completed tasks in this task bundling plan is ($\frac{77}{288}$ * 2) + $\frac{1}{3}$ = $\frac{125}{144}$.

In fact, this task bundling plan is the best bundling plan in this scenario. There are 4 task bundling plans besides this bundling plan. We name the four plans A, B, C, D, respectively. In plan A, each task will be regarded as a bundle with only one task, and the number of expected completed tasks in this bundle is $\frac{1}{3}$ * 1 + $\frac{1}{5}$ * 1 + $\frac{1}{8}$ * 1 = $\frac{79}{120}$. In plan B, we will bundle all the three tasks as a bundle and the number of expected completed task is $\frac{59}{102}$. In plan C, we will bundle task t_1 and t_2 as a bundle, then the number of expected completed task is $\frac{19}{40}$. Finally, in plan D, task t_1 and task t_3 will be bundled, and the number of expected completed task is $\frac{41}{75}$.

With the number of workers increases, the number of expected completed tasks calculated by the initial solution construction algorithm also increases and gradually approaches 3, which is the total number of all tasks.

Theorem 2: The time complexity of the initial solution construction algorithm is $O(n^2)$, where n is the number of tasks.

Proof: This algorithm includes three steps. The first step aims to sort the tasks based on the value of ALOW probability from large to small. We use bubble sorting algorithm as the sorting algorithm and the time complexity of bubble sorting algorithm is $O(n^2)$. The time complexity of the second step is $O(1)$, and the time complexity of the third step is $O(n)$. The second and third steps will be repeated at most n times, so the time complexity of the second and third step executing for at most n times is $O(n^2)$. As a result, the time complexity of the initial solution construction algorithm is $O(n^2)$, where n is the number of tasks.

The pseudo-code of the initial solution construction algorithm is shown in Algorithm 1, and we set “NECT” as the alias of "the number of expected completed tasks" in the pseudo-code.

6.2 | Simulated annealing optimization

Now we will use the simulated annealing optimization algorithm to optimize the initial solution obtained by the greedy based algorithm. The simulated annealing algorithm is presented in Algorithm 2. As the name suggests, this algorithm is based on simulated annealing. T_0, T^*, δ represent the initial temperature, the final temperature and the cooling rate, respectively. The main idea of the simulated annealing optimization algorithm is as follows: This algorithm runs to adjust the solution S_0 until $T_0 \leq T^*$. We define a temporary variable S and assign it an initial value S_0. Line 4 to line 26 is the main loop of the simulated annealing. In each iteration of the main loop, we do the following operations. First, we randomly select a task bundle b_1 from the solution S and randomly remove a task t from b_1. Second, we randomly select another task bundle b_2 from S and insert the task t into b_2, so that we get a new solution S'. Then we calculate the number of expected completed tasks of S (denoted as n_1) and the number of expected completed tasks of S' (denoted as n_2). If $n_2 > n_1$, then we assign S' to S. Otherwise, we randomly generate a number R whose value ranges from 0 to 1, next we calculate $e^{\frac{n_2-n_1}{T^*}}$, if $e^{\frac{n_2-n_1}{T^*}} > R$, we will assign S' to S. Thirdly, we update the value of temperature from T_0 to $T_0 * \delta$. We repeat the three steps until T_0 is not greater than T^*.

6.3 | Repeatedly simulated annealing

In order to optimize the solution obtained by the second phase, we can repeat the second phase for many times until the increase of the number of expected completed tasks approaches zero. Based on the initial solution construction algorithm and the simulated annealing algorithm, we give the pseudo-code of LocTrajBundling in Algorithm 3.

Algorithm 1 Initial solution construction

Require:
traffic network $G = (V, E)$,
worker set W,
task set T
Ensure:
initial task bundling plan S_0

Define two empty sets S_0, S_1;
Calculate the ALOW probability of each task;
sort the task set T to get a queue Q according to the value of ALOW probability from the largest to the smallest;
while Q is not empty **do**
    q' = the first element of Q;
    $S_1 = S_1 \bigcup q'$;
    remove q' from Q;
    if Q is empty **then**
        $S_0 = S_0 \bigcup \{S_1\}$;
        return S_0;
    end if
    q'' = the element that can maximize the increase of $NECT$ if we bundle this element with S_1;
    $increase$ = the increase of $NECT$ if we bundle q'' with S_1;
    if $increase \geq 0$ **then**
        $S_1 = S_1 \bigcup q''$;
        remove q'' from Q;
        if Q is empty **then**
            $S_0 = S_0 \bigcup \{S_1\}$;
            return S_0;
        end if
    else
        $S_1' = S_1$;
        $S_1 = \emptyset$;
        $S_0 = S_0 \bigcup \{S_1'\}$;
    end if
end while
return S_0;

7 | EVALUATION

To compare the performance of different algorithms, we give some definitions as follows:

Definition 7 (Task Bundling Performance Ratio(TBPR)). The task bundling performance result of an algorithm is that, for a given case, the average of the number of expected completed tasks (NECT) in each experiment. The task bundling performance ratio (TBPR) of an algorithm in a given case is the task bundling performance result of LocTrajBundling algorithm divided by the task bundling performance result of this algorithm when using the same data and running the same time experiments. TBPR is greater than 1 means that the task bundling plan produced by the algorithm can reach higher task bundling performance result than our algorithm, and the larger the value of TBPR, the better the performance of this algorithm in producing task bundling plan with higher task bundling performance result.

Definition 8 (Worker-Task Ratio (WTR)). The worker-task ratio is the ratio of the number of workers to the number of tasks.

Definition 9 (Task Completion Ratio (TCR)). Similar to TBPR, the task completion ratio of an algorithm is that, for a given case, the average of the number of expected tasks to the number of all the tasks in each experiment. The higher TCR value of an

Algorithm 2 Simulated annealing optimization

Require:
traffic network $G = (V, E)$,
worker set W,
task set T,
T_0, T^*, δ,
initial task bundling plan S_0

Ensure:
Improved task bundling plan S

$S = S_0$;
while $T_0 \geq T^*$ **do**
  $S_{temp} = S$;
  $b_1 =$ a randomly selected element of S;
  $S = S \setminus \{b_1\}$;
  $b_2 =$ a randomly selected element of S;
  $S = S \setminus \{b_2\}$;
  $t =$ a randomly selected element of b_1;
  $b_1 = b_1 \setminus \{t\}$;
  $b_2 = b_2 \bigcup \{t\}$;
  $S' = S \bigcup b_1 \bigcup b_2$;
  $S = S_{temp}$;
  $n_1 =$ the number of expected completed tasks of S;
  $n_2 =$ the number of expected completed tasks of S';
  if $n_2 > n_1$ **then**
    $S = S'$;
  else
    $R =$ a random number whose value ranges from 0(excluded) to 1(excluded);
    if $e^{\frac{n_2 - n_1}{T^*}} > R$ **then**
      $S = S'$;
    end if
  end if
  $T_0 = T_0 * \delta$;
end while
return S;

algorithm, the better the performance of this algorithm in producing task bundling plan with higher task bundling performance result.

Definition 10 (Worker Efficiency Performance (WEP)). The worker efficiency of a worker is the expected reward of him/her divided by the number of expected completed tasks of him/her. For workers, the higher worker efficiency, the greater the average reward of completing a task. The worker efficiency performance of an algorithm in a given case is the average of the average worker efficiency of all workers in each experiment.

We have done several experiments to test the performance of our algorithm in producing task bundling plan. Next, we will introduce other algorithms we used to compare with our algorithm, and we will also introduce the datasets with their experiment settings and the experimental results.

Algorithm 3 LocTrajBundling

Input:
traffic network $G = (V, E)$,
worker set W,
task set T,
T_0, T^*, δ
Output:
A final task bundling plan S_f

1: S_0 = Initial solution construction(G, W, T);
2: $increase = 1$;
3: **while** $increase > 0$ **do**
4: $previous$ = the number of expected completed tasks of S_0;
5: S_0 = Simulated annealing optimization ($G, W, T, T_0, T^*, \delta, S_0$);
6: new = the number of expected completed tasks of S_0;
7: $increase = new - previous$;
8: **end while**
9: $S_f = S_0$;
10: return S_f;

7.1 | Algorithms used for comparing with our LocTrajBundling algorithm

We design two algorithms called NoBundle, and Initial. We also implement two algorithms which are designed for the task assignment in requester-centric mobile crowdsensing (RCMCS) in[26] and[27], and we name the two algorithms LRBA and B-DBA, respectively. As we all know, the output of a task assignment algorithm for RCMCS is several "worker-task bundle" pairs. Therefore, we remove the information of workers from all the "work-task bundle" pairs and get a task bundling plan. We use the four algorithms to compare with our algorithm. The brief descriptions of the four algorithms are as follows.

Initial: In this algorithm, we will use the output of the first phase of our LocTrajBundling algorithm as the task bundling plan. The task bundling plan is S = Initial solution construction(G, W, T).

NoBundle: In this algorithm, no task will be bundled. The task bundling plan is $S = \{\{t_1\}, \{t_2\}, \ldots, \{t_n\}\}$.

LRBA: In[26], the authors formulate a problem named (MRP) which aims to maximize the rewards of the platform in RCMCS, where each worker has his/her own travel distance budget, and each task is location dependent. Then the authors design a task assignment algorithm named LRBA to solve this problem. We change the optimization objective from "maximize the rewards of the platform" to "maximize the number of expected completed tasks", and implement this algorithm.

B-DBA: In[27], the authors study the task assignment problem whose objective is maximizing the total task quality in RCMCS under the constraint of travel distance budget of mobile workers. The authors provide a Bio-inspired Travel-Distance-Balance-based algorithm (B-DBA) which is based on Pareto Ant Colony Optimization. To implement this algorithm, the optimization objective is changed from "maximize the total task quality" to "maximize the number of expected completed tasks".

7.2 | Synthetic Dataset

In the synthetic dataset experiments, we have to ensure that the vertices which are randomly generated should be located in Changchun city. For simplicity, in each time experiment, the latitude and longitude of each vertex are randomly generated in specific ranges. The longitudes range from 125.1 to 125.5 and the latitudes range from 43.8 to 44.

7.3 | Real-world Dataset

To do the real-world dataset experiments, we obtain the latitudes and longitudes of 400 hotels in Changchun as vertices and use BaiduMap Web API to calculate the road length between all vertices as our traffic network. We will give a detailed introduction about how we use BaiDuMap Web API to get these data.

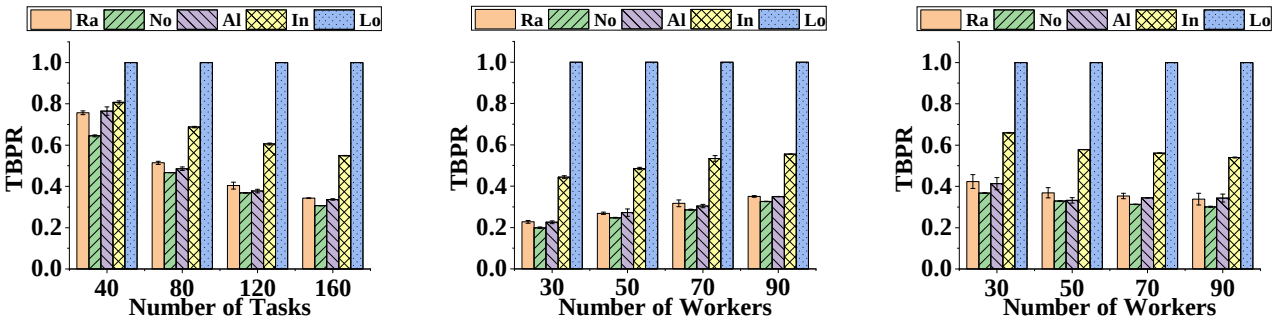
Step 1. Get the location of 400 hotels

In the real-world dataset experiments, we have to ensure that each vertex in the dataset corresponds to a real POI (Point of Interest) in Changchun. So we use place API, which is used to find a specific kind of POI in a given region, to find 400 hotels in Changchun. We randomly select the detailed information of two hotels in Changchun which will be used later. The detailed information of hotels mainly includes *name*, *location*, and *address*.

We will use the location information of each hotel at the step 2: get the road length between each pair of hotels.

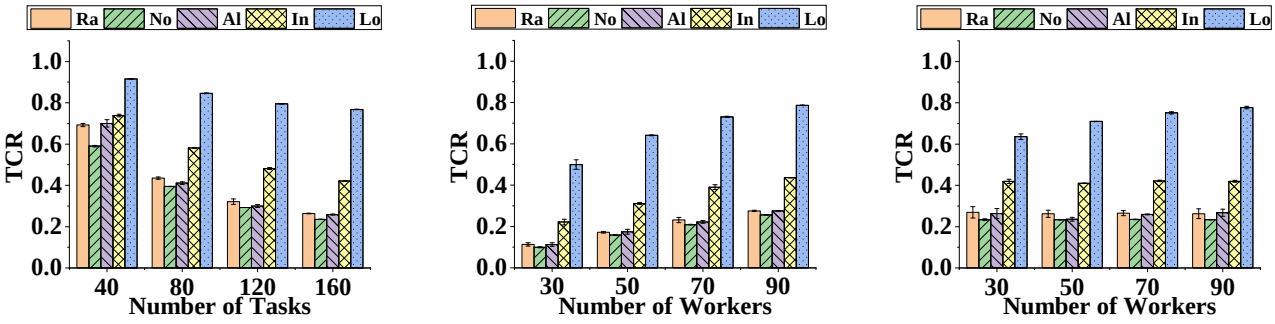
Step 2. Get the road length between each pair of hotels

We use Direction API to get the road length between each pair of hotels. The Direction API can search for qualified walking route plan based on the coordinates of the starting point and end point. The returned message contains an attribute named "distance", which represents the walking distance in this plan. We regard the attribute "distance" as the road length between two hotels. To use this API, we need to input the latitude and longitude of starting point A and end point B.



(a) The number of workers is fixed as 80 **(b)** The number of tasks is fixed as 160 **(c)** WTR is fixed as 0.5

FIGURE 4 TBPR comparisons in the real-world dataset experiments with three situations



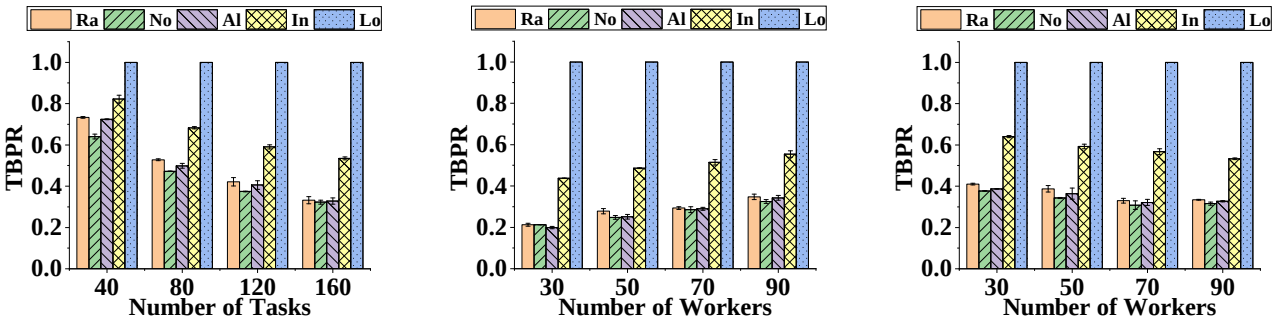
(a) The number of workers is fixed as 80 **(b)** The number of tasks is fixed as 160 **(c)** WTR is fixed as 0.5

FIGURE 5 TCR comparisons in the real-world dataset experiments with three situations

In every experiment where the number of tasks and the number of workers are given, we randomly select some vertices to formulate a set of tasks and a set of workers to do our experiment.

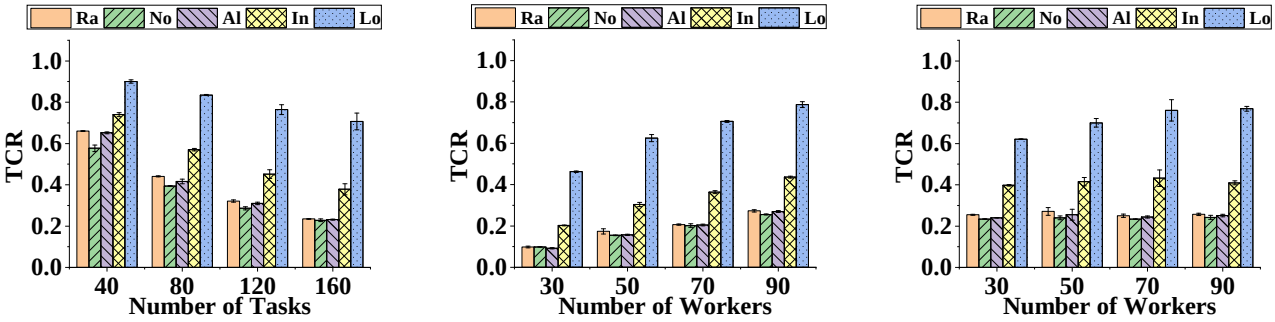
7.4 | Experiment settings

Settings for testing our LocTrajBundling algorithm: In both the real-world dataset experiments and the synthetic dataset experiments, we use three situations to test the performances of our algorithm. The first situation is that the number of workers is fixed as 80; the number of tasks is variable in different cases. The second situation is that the number of tasks is fixed as 160; the number of workers is variable in different cases. And in the third situation, the worker-task ratio is fixed as 0.5; the number of tasks and the number of workers is variable in different cases.



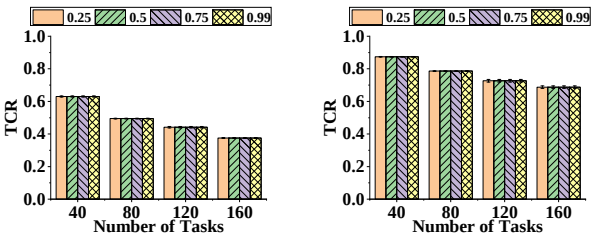
(a) The number of workers is fixed as 80 **(b)** The number of tasks is fixed as 160 **(c)** WTR is fixed as 0.5

FIGURE 6 TBPR comparisons in the synthetic dataset experiments with three situations



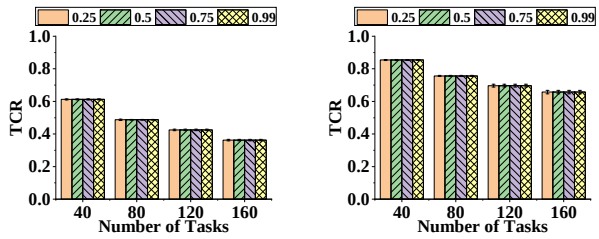
(a) The number of workers is fixed as 80 **(b)** The number of tasks is fixed as 160 **(c)** WTR is fixed as 0.5

FIGURE 7 TCR comparisons in the synthetic dataset experiments with three situations



(a) the number of workers is fixed as 20 **(b)** the number of workers is fixed as 60

FIGURE 8 TCR comparisons in the real-world dataset experiments with four cooling rates



(a) the number of workers is fixed as 20 **(b)** the number of workers is fixed as 60

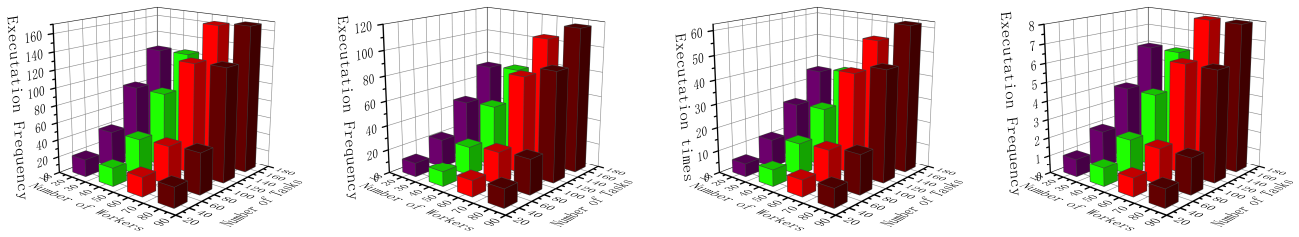
FIGURE 9 TCR comparisons in the synthetic dataset experiments with four cooling rates

A situation includes several cases, for example, in the first situation we mentioned above, there are cases such as 80 workers with 40 tasks, 80 workers with 80 tasks, and so on. In each case, we use different data and run 100000 times experiment, and then we calculate the average TBPR and TCR.

In our LocTrajBundling algorithm, T_0 is set as 10000, T^* is set as e^{-18}, and δ is set as 0.99 in all the situations of both the real-world dataset experiments and the synthetic dataset experiments.

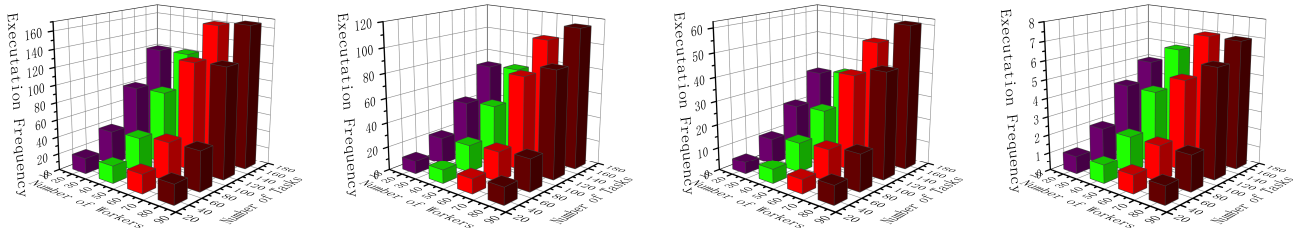
Settings for testing the impact of different parameters on our LocTrajBundling algorithm: There are totally 3 parameters in our LocTrajBundling algorithm: T_0 that represents the initial temperature, T^* that represents the final temperature, δ that represents the cooling rate. These three parameters jointly determine the number of iterations in the second phase of our LocTrajBundling algorithm, and the change of each parameter will affect the number of iterations in the second phase.

Essentially, the impact of the three parameters on our LocTrajBundling algorithm is the impact of the number of iterations in the second phase on our LocTrajBundling algorithm. For simplicity, we regard the change of the value of δ as the change of



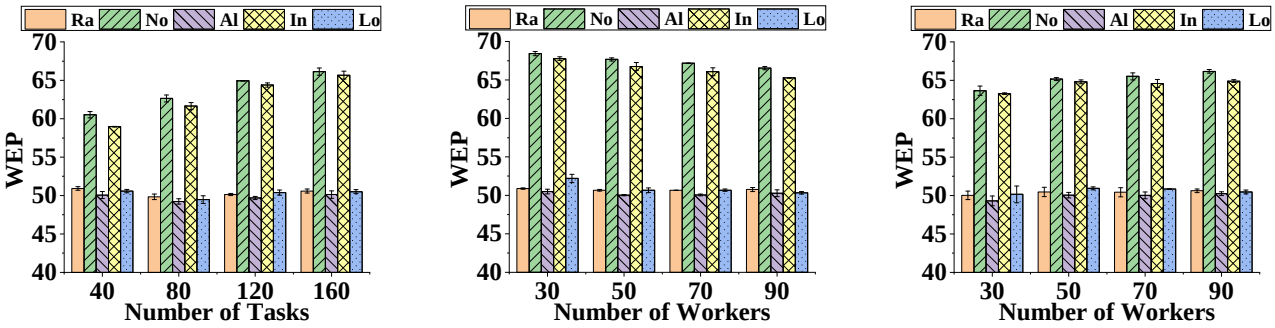
(a) the cooling rate is fixed as 0.25 **(b)** the cooling rate is fixed as 0.50 **(c)** the cooling rate is fixed as 0.75 **(d)** the cooling rate is fixed as 0.99

FIGURE 10 The execution frequency of the simulated annealing optimization algorithm in the real-world dataset experiments



(a) the cooling rate is fixed as 0.25 **(b)** the cooling rate is fixed as 0.50 **(c)** the cooling rate is fixed as 0.75 **(d)** the cooling rate is fixed as 0.99

FIGURE 11 The execution frequency of the simulated annealing optimization algorithm in the synthetic dataset experiments



(a) The number of workers is fixed as 80 **(b)** The number of tasks is fixed as 160 **(c)** WTR is fixed as 0.5

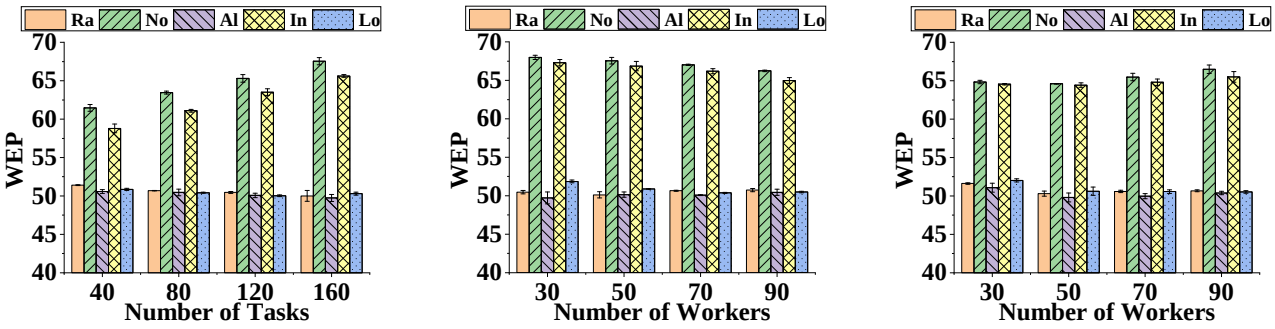
FIGURE 12 WEP comparisons in the real-world dataset experiments with three situations

the number of iterations in the second phase; we fix the value of T_0 as 10000 and fix the value of T^* as e^{-18}, and the value of δ ranges from 0.25 to 0.99.

We use four situations to test the impact of the number of iterations in the second phase on our LocTrajBundling algorithm, the values of δ in the four situations are 0.25, 0.5, 0.75, 0.99, respectively. In each situation, the number of workers ranges from 20 to 80, and the number of tasks ranges from 40 to 160. We use both the real-world dataset and the synthetic dataset to do experiments, and we use different data and run 100000 times experiment in each case.

7.5 | Results

Results about the TBPR of our LocTrajBundling algorithm: We show the experimental results of the real-world dataset experiments about the performance of our LocTrajBundling algorithm in Fig. 4 -(a) to Fig. 4 -(c) and Fig. 5 -(a) to Fig. 5 -(c), and we show the experimental results of the synthetic dataset experiments about the performance of our LocTrajBundling algorithm in Fig. 6 -(a) to Fig. 6 -(c) and Fig. 7 -(a) to Fig. 7 -(c). The “Ra”, “No”, “Al” , “In” and ‘Lo”in Fig. 4 to Fig. 7 represent the abbreviation of “LRBA”, “NoBundle”, “B-DBA” , “Initial” and “LocTrajBundling”, respectively. The error bars in Fig. 4 to Fig. 7 represent the value of standard deviation.



(a) The number of workers is fixed as 80 **(b)** The number of tasks is fixed as 160 **(c)** WTR is fixed as 0.5

FIGURE 13 WEP comparisons in the synthetic dataset experiments with three situations

As we can see, our algorithm can reach the highest TBPR among the five algorithms in all the six situations we mentioned before in both the real-world dataset experiments and the synthetic dataset experiments. Initial has the second best TBPR in all the three situations. In Fig. 4 -(a) and Fig. 6 -(a), the TBPR values of LRBA, NoBundle and B-DBA are closed to the TBPR value of Initial at first, but decrease rapidly with the number of tasks increases. In Fig. 4 -(b) and Fig. 6 -(b), when the number of tasks is fixed as 160, the TBPR values of the algorithms except our LocTrajBundling algorithm increase with the number of workers increases. In Fig. 4 -(c) and Fig. 6 -(c), with the number of workers increases, the TBPR values of the five algorithms remain stable as a whole when the WTR is fixed as 0.5.

Results about the TCR of our LocTrajBundling algorithm: As for TCR, we can see that our LocTrajBundling algorithm can also reach the highest TCR value among all the five algorithms in the three situations. In Fig. 5 -(a) and Fig. 7 -(a), the TCR value of our LocTrajBundling algorithm decreases with the increase of the number of tasks, but it is still higher than the TCR values of other four algorithms; the TCR values of LRBA, NoBundle and B-DBA are closed to the TCR value of Initial at first, but decrease rapidly with the number of tasks increases. In Fig. 5 -(b) and Fig. 7 -(b), when the number of tasks is fixed as 160, the TCR values of all the five algorithms increase with the number of workers increases. In Fig. 5 -(c) and Fig. 7 -(c), when the WTR is fixed as 0.5, the TCR value of our LocTrajBundling algorithm increases with the number of workers increases, while the TCR values of the other four algorithms remain stable as a whole.

Results about the impact of different parameters on our LocTrajBundling algorithm: In Fig. 8 -(a) to Fig. 8 -(b) and Fig. 9 -(a) to Fig. 9 -(b), we can see that, in both the real-world dataset experiments and the the synthetic dataset experiments, when the number of tasks and the number of workers are fixed, the change of the value of cooling rate δ has almost no impact on the TCR performance of our LocTrajBundling algorithm. It is because that, in our LocTrajBundling algorithm, the simulated annealing optimization algorithm will be repeatedly executed until the increase of the number of expected completed tasks approach zero, so that the output of our LocTrajBundling algorithm will approach a specific value when the number of tasks and the number of workers are fixed, no matter what the cooling rate. Therefore, it can be concluded that the change of the three parameters has almost no impact on the TCR performance of our LocTrajBundling algorithm.

However, despite that the change of the three parameters will not affect the TCR performance of our LocTrajBundling algorithm, it will affect the execution frequency of the simulated annealing optimization algorithm. The execution frequency of the simulated annealing optimization algorithm represents the number of times the simulated annealing optimization algorithm is executed. It is obvious that the performance of the simulated annealing optimization algorithm depends on the number of iterations, and a higher number of iterations leads to the better performance of the simulated annealing optimization algorithm. The better performance of the simulated annealing optimization algorithm, the less execution frequency of the simulated annealing optimization algorithm. In Fig. 10 -(a) to Fig. 10 -(d) and Fig. 11 -(a) to Fig. 11 -(d), we can see that, the execution frequency of the simulated annealing optimization algorithm decreases rapidly with the cooling rate δ increases when the number of tasks, and the number of workers are fixed in both the real-world dataset experiments and the synthetic dataset experiments.

Results about the WEP of our LocTrajBundling algorithm: From Fig. 12 -(a) to Fig. 12 -(c) and Fig. 13 -(a) to Fig. 13 -(c), we can see that, in both the real-world dataset experiments and the the synthetic dataset experiments, our LocTrajBundling algorithm can not reach the highest WEP value among all the five algorithms in the three situations. This is because our LocTrajBundling algorithm aims to maximize the benefit of requester in WCMCS by task bundling. However, task bundling will reduce the freedom of workers to select tasks, so that workers may have to complete some tasks they don't want to complete and suffer benefit damage. In all situations of both the real-world dataset experiments and the the synthetic dataset experiments,

the experimental results show that NoBundle algorithm reaches higher WEP value than the other four algorithms because the workers in NoBundle have complete freedom in selecting tasks.

8 | CONCLUSION

MCS is a promising sensing paradigm and has developed rapidly in recent years. Task assignment is a crucial problem in mobile crowdsensing and has been studied for many years. Most existing researches mainly concentrate on the task assignment problem in requester-centric mobile crowdsensing (RCMCS), where the requester assigns tasks to workers, and workers passively complete the tasks assigned by the requester. Few researches focus on worker-centric mobile crowdsensing (WCMCS), where workers can determine what tasks they will complete to maximize their benefits. In WCMCS, workers are more likely to select the tasks which are nearer to their trajectories or the tasks with higher rewards due to the individual rationality, so that some tasks may be selected by numerous workers while some tasks may be selected by no workers. Therefore, it is hard to achieve global optimum (maximize the number of expected completed tasks) in WCMCS where we empower workers autonomously select tasks to complete, so that the requester in WCMCS will suffer benefit damage because the requester wants to maximize the number of expected tasks. In this paper, we apply task bundling to address this issue and formulate a novel task bundling problem in worker-centric mobile crowdsensing with the objective of maximizing the number of expected completed tasks. In order to solve this problem, we propose a heuristic algorithm named LocTrajBundling with two phases: initial solution construction and simulated annealing optimization. The LocTrajBundling algorithm bundles tasks according to the location of tasks and the trajectories of workers. Compared with other algorithms, experimental results show that, in both the real-world dataset experiments and the synthetic dataset experiments, our algorithm can reach the highest task bundling performance ratio in contrast with other four algorithms named LRBA, NoBundle, B-DBA and Initial.

ACKNOWLEDGMENTS

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the National Natural Science Foundations of China under Grant No. 61772230, No. 61972450 and No. 62072209, Natural Science Foundations of Jilin Province No. 20190201022JC, National Science Key Lab Fund Project No. 61421010418, Innovation Capacity Building Project of Jilin Province Development and Reform Commission No. 2020C017-2, Changchun Science and Technology Development Project No.18DY005, Key Laboratory of Defense Science and Technology Foundations No. 61421010418, Jilin Province Young Talents Lifting Project No. 3D4196993421, and in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, and CNS 1564128.

Conflict of interest

The authors declare no potential conflict of interests.

References

1. Ganti RK, Ye F, Lei H. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine* 2011; 49(11): 32-39.

2. Khan WZ, Xiang Y, Aalsalem MY, Arshad Q. Mobile Phone Sensing Systems: A Survey. *IEEE Communications Surveys Tutorials* 2013; 15(1): 402-427.

3. Guo B, Wang Z, Yu Z, Wang Y, Zhou X. Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm. *Acm Computing Surveys* 2015; 48(1). doi: 10.1145/2794400

4. Ullah F, Al-Turjman F, Nayyar A. IoT-based green city architecture using secured and sustainable android services. *Environmental Technology & Innovation* 2020; 20: 101091.

5. Gupta H, Bhardwaj D, Agrawal H, Tikkiwal VA, Kumar A. An IoT Based Air Pollution Monitoring System for Smart Cities. In: *2019 IEEE International Conference on Sustainable Energy Technologies and Systems (ICSETS).* ; 2019: 173-177. doi: 10.1109/ICSETS.2019.8744949

6. Dutta J, Gazi F, Roy S, Chowdhury C. AirSense: Opportunistic crowd-sensing based air quality monitoring system for smart city. In: *2016 IEEE SENSORS.* ; 2016: 1-3. doi: 10.1109/ICSENS.2016.7808730

7. Zhang R, Liu W, Jia Y, Jiang G, Xing J, Jiang H, Liu J. WiFi Sensing-Based Real-Time Bus Tracking and Arrival Time Prediction in Urban Environments. *IEEE Sensors Journal* 2018; 18(11): 4746-4760.

8. Zhou P, Zheng Y, Li M. How Long to Wait? Predicting Bus Arrival Time With Mobile Phone Based Participatory Sensing. *IEEE Transactions on Mobile Computing* 2014; 13(6): 1228-1241.

9. Wang X, Zhang J, Tian X, Gan X, Guan Y, Wang X. Crowdsensing-Based Consensus Incident Report for Road Traffic Acquisition. *IEEE Transactions on Intelligent Transportation Systems* 2018; 19(8): 2536-2547.

10. Peng Z, Gao S, Xiao B, Guo S, Yang Y. CrowdGIS: Updating Digital Maps via Mobile Crowdsensing. *IEEE Transactions on Automation Science and Engineering* 2018; 15(1): 369-380.

11. Wang E, Zhang M, Cheng X, Yang Y, Liu W, Yu H, Wang L, Zhang J. Deep Learning-Enabled Sparse Industrial CrowdSensing and Prediction. *IEEE Transactions on Industrial Informatics* 2020: 1-1. doi: 10.1109/TII.2020.3028616

12. Gao Y, Dong W, Guo K, Liu X, Chen Y, Liu X, Bu J, Chen C. Mosaic: A low-cost mobile sensing system for urban air quality monitoring. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications.* ; 2016: 1-9. doi: 10.1109/INFOCOM.2016.7524478

13. Wan J, Liu J, Shao Z, Vasilakos A, Imran M, Zhou K. Mobile Crowd Sensing for Traffic Prediction in Internet of Vehicles. *Sensors* 2016; 16(1): 88. doi: 10.3390/s16010088

14. Yin X, Liu J, Cheng X, Zeng B, Xiong X. A low-complexity design for the terminal device of the urban IoT-oriented heterogeneous network with ultra-high-speed OFDM processing. *Sustainable Cities and Society* 2020; 61: 102323. doi: https://doi.org/10.1016/j.scs.2020.102323

15. Gupta O, Goyal N, Anand D, Kadry S, Nam Y, Singh A. Underwater Networked Wireless Sensor Data Collection for Computational Intelligence Techniques: Issues, Challenges, and Approaches. *IEEE Access* 2020; 8: 122959-122974.

16. Blair SJ, Bi Y, Mulvenna MD. Aggregated topic models for increasing social media topic coherence. *Applied Intelligence* 2020; 50(1): 138–156.

17. Liu W, Yang Y, Wang E, Wu J. User Recruitment for Enhancing Data Inference Accuracy in Sparse Mobile Crowdsensing. *IEEE Internet of Things Journal* 2020; 7(3): 1802-1814.

18. Wang E, Yang Y, Lou K. User selection utilizing data properties in mobile crowdsensing. *Information Sciences* 2019; 490: 210 - 226.

19. Xiao M, Wu J, Huang L, Wang Y, Liu C. Multi-task assignment for crowdsensing in mobile social networks. In: *2015 IEEE Conference on Computer Communications (INFOCOM).* ; 2015: 2227-2235. doi: 10.1109/INFOCOM.2015.7218609

20. Xiong H, Zhang D, Chen G, Wang L, Gauthier V, Barnes LE. iCrowd: Near-Optimal Task Allocation for Piggyback Crowdsensing. *IEEE Transactions on Mobile Computing* 2016; 15(8): 2010-2022.

21. Abdel-Basset M, Mohamed R, Elhoseny M, Bashir AK, Jolfaei A, Kumar N. Energy-Aware Marine Predators Algorithm for Task Scheduling in IoT-based Fog Computing Applications. *IEEE Transactions on Industrial Informatics* 2020: 1-1. doi: 10.1109/TII.2020.3001067

22. Deng D, Shahabi C, Demiryurek U, Zhu L. Task Selection in Spatial Crowdsourcing from Worker's Perspective. *Geoinformatica* 2016; 20(3): 529–568. doi: 10.1007/s10707-016-0251-4

23. Wu L, Du X, Zhang H, Yu W, Wang C. Effective task scheduling in proximate mobile device based communication systems. In: *2015 IEEE International Conference on Communications (ICC).* ; 2015: 3503-3508. doi: 10.1109/ICC.2015.7248867

24. Orouskhani M, Shi D, Cheng X. A Fuzzy Adaptive Dynamic NSGA-II With Fuzzy-Based Borda Ranking Method and its Application to Multimedia Data Analysis. *IEEE Transactions on Fuzzy Systems* 2021; 29(1): 118-128. doi: 10.1109/TFUZZ.2020.2979119

25. He S, Shin D, Zhang J, Chen J, Lin P. An Exchange Market Approach to Mobile Crowdsensing: Pricing, Task Allocation, and Walrasian Equilibrium. *IEEE Journal on Selected Areas in Communications* 2017; 35(4): 921-934.

26. He S, Shin D, Zhang J, Chen J. Toward optimal allocation of location dependent tasks in crowdsensing. In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications.* ; 2014: 745-753. doi: 10.1109/INFOCOM.2014.6848001

27. Gong W, Zhang B, Li C. Location-Based Online Task Assignment and Path Planning for Mobile Crowdsensing. *IEEE Transactions on Vehicular Technology* 2019; 68(2): 1772-1783. doi: 10.1109/TVT.2018.2884318

28. Wang L, Zhang D, Pathak A, Chen C, Xiong H, Yang D, Wang Y. CCS-TA: Quality-Guaranteed Online Task Allocation in Compressive Crowdsensing. In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ; 2015: 683–694. doi: 10.1145/2750858.2807513

29. Wang L, Zhang D, Wang Y, Chen C, Han X, M'hamed A. Sparse mobile crowdsensing: challenges and opportunities. *IEEE Communications Magazine* 2016; 54(7): 161-167.

30. Xu G, Li X, Jiao L, Wang W, Liu A, Su C, Zheng X, Liu S, Cheng X. BAGKD: A Batch Authentication and Group Key Distribution Protocol for VANETs. *IEEE Communications Magazine* 2020; 58(7): 35-41. doi: 10.1109/MCOM.001.2000118

31. Zhang M, Yang P, Tian C, Tang S, Gao X, Wang B, Xiao F. Quality-Aware Sensing Coverage in Budget-Constrained Mobile Crowdsensing Networks. *IEEE Transactions on Vehicular Technology* 2016; 65(9): 7698-7707.

32. Karaliopoulos M, Telelis O, Koutsopoulos I. User recruitment for mobile crowdsensing over opportunistic networks. In: *2015 IEEE Conference on Computer Communications (INFOCOM).* ; 2015: 2254-2262. doi: 10.1109/INFOCOM.2015.7218612

33. Wang E, Yang Y, Wu J, Liu W, Wang X. An Efficient Prediction-Based User Recruitment for Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* 2018; 17(1): 16-28. doi: 10.1109/TMC.2017.2702613

34. Wang L, Yang D, Han X, Wang T, Zhang D, Ma X. Location Privacy-Preserving Task Allocation for Mobile Crowdsensing with Differential Geo-Obfuscation. In: *Proceedings of the 26th International Conference on World Wide Web.* ; 2017: 627–636. doi: 10.1145/3038912.3052696

35. Pournajaf L, Garcia-Ulloa DA, Xiong L, Sunderam V. Participant Privacy in Mobile Crowd Sensing Task Management: A Survey of Methods and Challenges. *SIGMOD Rec.* 2016; 44: 23–34.

36. Abadi A, Terzis S, Metere R, Dong C. Efficient Delegated Private Set Intersection on Outsourced Private Datasets. *IEEE Transactions on Dependable and Secure Computing* 2019; 16(4): 608-624.

37. Wang X, Qin X, Hosseini MB, Slavin R, Breaux TD, Niu J. GUILeak: Tracing Privacy Policy Claims on User Input Data for Android Applications. In: *ICSE '18.* Association for Computing Machinery; 2018; New York, NY, USA: 37–47. doi: 10.1145/3180155.3180196

38. Shankar A, Pandiaraja P, Sumathi K, Stephan T, Sharma P. Privacy preserving E-voting cloud system based on ID based encryption. *Peer-to-Peer Networking and Applications* 2020: 1–11. doi: 10.1007/s12083-020-00977-4

39. Li J, Ye H, Li T, Wang W, Lou W, Hou T, Liu J, Lu R. Efficient and Secure Outsourcing of Differentially Private Data Publishing with Multiple Evaluators. *IEEE Transactions on Dependable and Secure Computing* 2020: 1-1.

40. Cheung MH, Hou F, Huang J, Southwell R. Distributed Time-Sensitive Task Selection in Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* 2020: 1-1. doi: 10.1109/TMC.2020.2975569

41. Wang J, Wang L, Wang Y, Zhang D, Kong L. Task Allocation in Mobile Crowd Sensing: State-of-the-Art and Future Opportunities. *IEEE Internet of Things Journal* 2018; 5(5): 3747-3757.

42. Zhang D, Xiong H, Wang L, Chen G. CrowdRecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ; 2014: 703–714. doi: 10.1145/2632048.2632059

43. Liu Y, Guo B, Wang Y, Wu W, Yu Z, Zhang D. TaskMe: Multi-Task Allocation in Mobile Crowd Sensing. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ; 2016: 403–414. doi: 10.1145/2971648.2971709

44. Liu CH, Zhang B, Su X, Ma J, Wang W, Leung KK. Energy-Aware Participant Selection for Smartphone-Enabled Mobile Crowd Sensing. *IEEE Systems Journal* 2017; 11(3): 1435-1446.

45. Xu J, Rao Z, Xu L, Yang D, Li T. Incentive Mechanism for Multiple Cooperative Tasks with Compatible Users in Mobile Crowd Sensing via Online Communities. *IEEE Transactions on Mobile Computing* 2020; 19(7): 1618-1633.

46. Phuttharak J, Loke SW. A Review of Mobile Crowdsourcing Architectures and Challenges: Toward Crowd-Empowered Internet-of-Things. *IEEE Access* 2019; 7: 304-324.

47. Yuen MC, King I, Leung KS. Taskrec: A task recommendation framework in crowdsourcing systems. *Neural Processing Letters* 2015; 41(2): 223–238.

48. Kandappu T, Jaiman N, Tandriansyah R, Misra A, Cheng SF, Chen C, Lau HC, Chander D, Dasgupta K. TASKer: Behavioral Insights via Campus-Based Experimental Mobile Crowd-Sourcing. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ; 2016: 392–402. doi: 10.1145/2971648.2971690

49. Wang Z, Hu J, Wang Q, Lv R, Wei J, Chen H, Niu X. Task Bundling Based Incentive for Location-Dependent Mobile Crowdsourcing. *IEEE Communications Magazine* 2019; 57(3): 132-137.

50. Hemmecke R, Köppe M, Lee J, Weismantel R. Nonlinear Integer Programming. *50 Years of Integer Programming* 2010: 561–618.

51. Aigner M. A Characterization of the bell numbers. *Discrete Mathematics* 1999; 205(1-3): 207-210.