Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7218061, 10 pages
https://doi.org/10.1155/2018/7218061

Research Article

Multitask Allocation to Heterogeneous Participants in Mobile Crowd Sensing

Weiping Zhu,[1] **Wenzhong Guo**,[1,2,3] **Zhiyong Yu**,[1,2] **and Haoyi Xiong**[4]

[1]*College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China*
[2]*Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China*
[3]*Key Laboratory of Spatial Data Mining and Information Sharing, Ministry of Education, Fuzhou 350003, China*
[4]*Department of Computer Science, Missouri University of Science and Technology, MO 65409, USA*

Correspondence should be addressed to Wenzhong Guo; guowenzhong@fzu.edu.cn

Received 10 March 2018; Revised 21 April 2018; Accepted 9 May 2018; Published 14 June 2018

Academic Editor: Huaqun Wang

Copyright © 2018 Weiping Zhu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Task allocation is a key problem in Mobile Crowd Sensing (MCS). Prior works have mainly assumed that participants can complete tasks once they arrive at the location of tasks. However, this assumption may lead to poor reliability in sensing data because the heterogeneity among participants is disregarded. In this study, we investigate a multitask allocation problem that considers the heterogeneity of participants (i.e., different participants carry various devices and accomplish different tasks). A greedy discrete particle swarm optimization with genetic algorithm operation is proposed in this study to address the abovementioned problem. This study is aimed at maximizing the number of completed tasks while satisfying certain constraints. Simulations over a real-life mobile dataset verify that the proposed algorithm outperforms baseline methods under different settings.

1. Introduction

An era of "Internet of Things" has been reached given the development of wireless communication, sensor technology, smartphones, and wearable devices. A new sensing paradigm called Mobile Crowd Sensing (MCS), where mobile devices play an important role in large-scale sensing and information sharing, has become research issue in academia and industry. Due to its advantages such as low cost and wide spatiotemporal coverage, MCS applications have been studied like intelligent transportation [1, 2], environment monitoring [3], target identification [4], and so on.

In contrast to traditional wireless sensor networks (WSN) [5], MCS is a human-centered sensing model, in which MCS applications must recruit participants to complete the sensing tasks. A straightforward way for obtaining a highly reliable sensing data is to recruit as many participants as possible to complete tasks. However, this strategy results in a high sensing cost. A key issue is allocating the tasks to proper participants, while accounting for the various initial locations of different participants, sensing data reliability, and sensing cost. Therefore, task allocation is an important issue in linebreak MCS.

Due to the importance, several studies have been conducted to select participants to complete tasks [6–20]. In these schemes, participants are selected to achieve a specific goal (e.g., maximize the regional coverage) under certain constraints, such as the budget cost. These studies have assumed that every user is indiscriminate; that is, everyone can complete any task provided that this individual arrives at the location of tasks. This assumption is reasonable in the initial stage of crowd sensing, where sensing tasks are simple and everyone can complete any type of task.

However, sensing tasks have become complicated with the increase in demands of crowd sensing. The abovementioned assumption may be invalid. On one hand, several tasks can be rather complex and not easily completed by participants. The platform requires that participants must preassemble special sensors or possess certain skills to accomplish tasks. On the other hand, different devices integrated into various sensors, which can accomplish different tasks. To complete complex tasks, it is imperative for platform to select participants whose devices preassemble the required sensors.

In this study, we consider the multitask allocation to heterogeneous participants (MTHP). The tasks may require the participants, whose devices preassemble the required sensors, to move to the location of tasks at a given time and complete these tasks. From the perspective of participants, everyone differs in sensing capability, including sensor type and the number of tasks that they can completed. There are two challenges that must be considered to overcome this problem. First, we must consider the tasks that the participants can complete during a specific duration in accordance with the sensors that participants carry and the location of participants and tasks, but only a few participants exist for all tasks. Second, from perspective of platform, it expects as many tasks as possible to be allocated, although several constraints may prevent the completion of the tasks. These constraints include the maximum number of tasks that participants can complete and the distance between the location of tasks and their current location. So the other challenge is on coordinating with participants to complete different tasks. Thus, we explore the multitask allocation problem by considering the heterogeneity of participants. The main contributions of this study are as follows:

(1) We formulate the problem of MTHP, which considers the heterogeneity among participants. Specifically, participants are different in sensor type, maximum workload, and moving speed. The object of MTHP is to maximize the number of accomplished tasks under the sensing capacity and time constraints.

(2) A greedy discrete particle swarm optimization with genetic algorithm (GDPSOGA) operation is specifically designed to tackle the multitask allocation problem. It first selects participants using heuristic strategies to reduce the search space of discrete particle swarm optimization (DPSO). Then, the random two-point mutation and random two-point crossover operations in genetic algorithm (GA) are incorporated to coordinate the participant resource among tasks and further maximize the number of completed tasks.

(3) Experiments on a real-world mobile usage dataset indicate that the proposed algorithm outperforms baseline methods under different settings.

The remainder of this paper is organized as follows: in Section 2, the related works are presented. In Section 3, the problem of MTHP is described in detail, and then our proposed algorithm and strategy are introduced in Section 4. In Section 5, we compare our algorithm with baseline algorithms and evaluate the performance of proposed algorithm. Finally, conclusions drawn from this study are presented in Section 6.

2. Related Work

Task allocation is a key research issue in MCS and has drawn considerable attention from researchers. In [6, 7], Reddy et al. considered the location, time constraints, and habits of users and proposed a coverage-based framework to select proper participants to maximize spatial coverage. Xiong et al. [8] defined a temporal-spatial coverage called k-depth coverage and then discussed selection of participants to maximize coverage under the constraints of budget and minimize the budget while satisfying the predefined coverage goal. Zhang et al. [9] investigated coverage quality and selected a subset of mobile users to maximize coverage quality under constrained budget. Several researchers considered selecting participants to minimize cost while guaranteeing data quality. Karaliopoulos et al. [10] studied the manner for selecting mobile users to minimize cost while ensuring the coverage of points of interest. Zhang et al. [11] predicted the mobility of participants and then selected minimum number of participants to meet the predefined temporal-spatial coverage. Wang et al. [12] proposed a framework that considers the spatial and temporal correlations among different subareas to reduce the number of participants required. Another work [13] defined a new coverage metric, namely, “t-sweep k-coverage”, and proposed two methods for selecting smallest set of candidate participants to satisfy predefined requirements. However, these works studied the task allocation problem for single task and have disregarded the competition of participants for sensing tasks.

Recently, several works have been proposed for the multitask allocation. Li et al. [14] proposed a greedy-based participant selection algorithm for heterogeneous tasks to minimize the number of participants while guaranteeing a certain level of coverage. Liu et al. [15] studied the multitask allocation problem under the two conditions. The first is few participants and many tasks. The second is many participants and few tasks. Wang et al. [16] considered the maximum workload of participants and proposed a two-phase offline multitask allocation approach. In [17], the goal is to select a subset of participants to maximize the quality of information under budget constraints. Guo et al. [18] proposed a worker selection framework for time-sensitive and delay-tolerant tasks. Time-sensitive tasks are aimed at minimizing the distance traveled by workers. For delay-tolerant tasks, the goal is to minimize the total number of workers. He et al. [19] studied the optimal allocation algorithms for location-dependent tasks to maximize the reward of platform. In [20], a novel framework was proposed to improve the data accuracy of task allocation with the aid of fog-nodes.

The related works presented above have mainly assumed that participants can accomplish tasks as long as they arrive at the location of tasks and ignored the heterogeneity among participants. For several complicated tasks, platform must select participants who already preassembled specific sensors to get reliable sensing data. In our study, we propose a task allocation framework that considers the heterogeneity among participants. In particular, we study the problem of multitask allocation toward heterogeneous participants, which assumes that everyone carries a mobile device integrated with different sensors and can accomplish different types of task. We present the modified PSO algorithm to address it.

3. Problem Formulation

In this section, we present the formal definition of the MTHP. Assume that the union of sensors is denoted as $S = \{s_1, s_2, s_3, \ldots, s_k\}$. Each user possesses one or multiple sensors in S. There are m users on the platform, which are denoted by the set $U = \{u_1, u_2, u_3, \ldots, u_m\}$. Each user can be depicted by a tuple: $(\mathrm{id}_i, \mathrm{loc}_i, S_i, v_i, q_i)$. It indicates that user u_i has a set S_i $(\subseteq S)$ of sensors, is located at position loc_i, and can move with velocity v_i anywhere. Due to the limited sensing capability, we assume that u_i can complete at most q_i tasks. Giving n different tasks on the platform are denoted by the set $T = \{t_1, t_2, t_3, \ldots, t_n\}$. Each task can be depicted by a tuple of five elements: $(\mathrm{loc}_j, s_j, p_j, st_j, et_j)$. To ensure the quality of sensing data, p_j users who preload sensor s_j are required to move to the location loc_j to complete the task during the time interval $[st_j, et_j]$. We use $T_{u_i} = \{t_1, t_2, t_3, \ldots\}$ to denote the task set that is allocated to u_i and $U_{t_j} = \{u_1, u_2, \ldots\}$ to denote the user set that completes the task t_j. The MTHP problem aims to allocate the tasks to the subset of the user set so that the number of tasks that can be accomplished is maximized.

To complete tasks, users must travel from their current location to the location of the tasks. We use Manhattan distance [21] to measure the distance between the location of the tasks and the users.

Based on the abovementioned definition, the MTHP problem can be formulated as

$$\max \quad \sum_{i=1}^{m} \left|T_{u_i}\right|$$
$$\text{s.t.} \quad \begin{cases} \left|T_{u_i}\right| \le q_i, & 1 \le i \le m \\ \left|U_{t_j}\right| = p_j, & 1 \le j \le n \\ S_{t_j} \in S_{u_i}, & 1 \le i \le m,\ 1 \le j \le n \\ st_j \le \dfrac{\mathrm{dis}\left(\mathrm{loc}_{u_i}, \mathrm{loc}_{t_j}\right)}{v_i} \le et_j, & u_i \in U_{t_j}, \end{cases} \tag{1}$$

where $\mathrm{dis}(\mathrm{loc}_{u_i}, \mathrm{loc}_{t_j})$ is the Manhattan distance between the task j and the user i.

This is a combinatorial optimization problem, and the solution space is quite large. For example, given n tasks and m users that satisfy all the constraints on the platform, if every task requires p participants, then there are $(C_m^p)^n$ combination schemes, and the value sharply increases with the increase of n, m, and p. Thus a heuristic task allocation algorithm must be designed to reduce the search space of the problem. Furthermore, considering the limited sensing capacity of participants, MTHP not only allocates tasks to users under the various constraints, but also coordinates users among tasks to further maximize the number of completed tasks, which makes the task allocation more complex. Several efficient coordination strategies must be introduced to achieve optimal utilization of participants and maximize the number of completed tasks. According to the analysis above, we adopt the greedy DPSO algorithm with GA (GDPSOGA) operation to solve this problem.

4. Algorithm

4.1. Basic Particle Swarm Optimization. Particle swarm optimization (PSO) is a population-based intelligence algorithm that simulates the movement of a flock of birds to seek food. It is a popular optimization method in many fields due to its simplicity and fast convergence. For PSO, a particle is defined as a potential solution to a problem in the D-dimensional space. Each particle adjusts its position and velocity according to its own experience and that of neighboring particles during the iteration. The particles are manipulated according to the following formula:

$$v_i^{t+1} = w \times v_i^t + c_1 r_1 \left(p_i - x_i^t\right) + c_2 r_2 \left(p_g - x_i^t\right) \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{3}$$

where t is the iteration index; v_i^t and x_i^t represent the velocity and position of particle i in the iteration t, respectively. p_i and p_g represent the previous optimal position of particle i and optimal position of all particles after $t-1$ iterations, respectively. w is the inertia weight, which indicates the impact of the last iteration to the current iteration. c_1 and c_2 are the acceleration factors, which reflect the simulative ability to the previous optimal solution and global optimal solution. r_1 and r_2 are random numbers distributed uniformly on the interval from 0 to 1.

Problem (1) is a discrete problem. Thus, the standard PSO is not appropriate for this problem due to its continuous nature. Inspired by our previous work [22], a DPSO is designed here to solve the problem.

4.2. Particle Representation. In the scenario of task allocation, each particle represents a potential allocation scheme for all tasks. A favorable particle representation considers not only the redundancy of the search space, but also the efficiency of the algorithm. According to [22], the particle encoding scheme should follow three main principles, that is, nonredundancy, completeness, and soundness.

Definition 1 (nonredundancy). A one-to-one relationship exists between the encoding scheme and the potential solution in the problem space.

Definition 2 (completeness). All feasible solutions can be represented by the particle according the encoding scheme.

Definition 3 (soundness). Each particle in the encoding space must correspond to the potential solution in the problem space.

Satisfying all three principles simultaneously is difficult for an encoding scheme. The task allocation problem is selecting users to complete tasks, so we adopt the task-user encoding scheme to represent a particle. Each point of particle indicates the user that is selected to complete corresponding task. For example, in Figure 1, four tasks are presented, and every task requires three participants. t_1 is allocated to u_1, u_2, and u_4, and other tasks are allocated similar to t_1.

Properties. The task-user encoding scheme satisfies the principles of completeness and soundness but does not meet the principle of nonredundancy.

```
Input: Task set $T$, User set $U$, the number of users that each task required $p$
Output: The <task, users> tuple
(1) For each task $t$ in $T$
(2)     Find the users with the required sensor and maximum workload > 0, denotes as $U_t$
(3)     Sort the $U_t$ according to the moving time, maximum workload and number of sensors
(4)     Greedily select users from $U_t$ whose moving time is shortest
(5)     If moving time is equal, select non-competitive users firstly
(6)     If all users are competitive users, select users whose devices with fewer sensors
(7)     Delete task $t$ if there are $p$ users to complete it
(8) End for
(9) Output the <task, users> tuple
```

Algorithm 1: Greedy strategy algorithm.

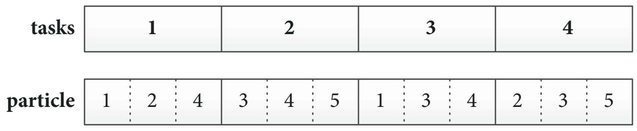


Figure 1: Particle encoding.

Proof. Each point of a particle represents a potential task allocation of corresponding task, so all feasible solutions of tasks can be represented by the particle according to the task-user encoding scheme; thus the task-user encoding scheme satisfies the principle of completeness. The problem optimizes the number of completed tasks, and the result can be computed through particle. So every particle corresponds to a potential solution in the problem space; i.e., the task-user encoding scheme satisfies the principle of soundness. However, the problem only counts the number of completed tasks, and different particles may correspond to the same result, although the solutions are different. Thus, the task-user encoding scheme does not satisfy the principle of nonredundancy. □

4.3. Fitness Function. Fitness function is introduced to evaluate the accuracy of each individual in achieving the goal of a problem. As discussed in Section 3, from perspective of platform, the optimization goal of the MTHP problem is to allocate tasks as many as possible. So we define the fitness function to count the number of completed tasks, as shown in formula (4).

$$\text{fitness} = \left|T_{u_i}\right| \tag{4}$$

4.4. Particle Initialization. For the PSO, the initial state of particle significantly influences the result because the final solution is derived from the initial solution. In general, the particle should be randomly initialed from the potential solution space. However, a random initialization particle may be far away from the optimal solution due to the huge solution space. Thus, several heuristic strategies are required for initialization processing.

For the MTHP problem, we attempt to maximize the number of completed tasks within the tasks during time. Inspired by work [18], we design a greedy strategy to select users. We firstly select the users that preload the sensor that the task required and move to the location of tasks within the shortest possible time. We give the priority to moving time to ensure that tasks can be completed while users have more extra time to complete other tasks.

When selecting candidate users according to the moving time, some of these users may be also required by other tasks simultaneously. In this case, we should consider the situation of maximum workload. Inspired by work [16], we define a user as competitive if the number of tasks that tend to select the user is greater than the maximum workload of the user. Otherwise we define the user as noncompetitive. We firstly consider the noncompetitive user and then leave the competitive user to other tasks to maximize the utilization of users.

Furthermore, the number of sensors that users carry must be considered. If all candidates are competitive users, we select the user with fewer sensors and leave users with more sensors to other types of tasks. The pseudocode is presented in Algorithm 1.

For example, as Tables 1 and 2 show, there are four tasks on the platform. The sensor they need is depicted as A, B, C, and D, respectively. Each task requires two different users to complete and only users carry the corresponding type of sensor can complete the task. There are three users $u_1,\ u_2,\ u_3$ on the platform. The circumstance of sensors and maximum workload are shown in Table 2. For simplicity, we assume that the moving time of users is equal and the task allocation depends on the maximum workload and sensors of users. Firstly, we consider the allocation of task t_1, u_1 and u_3 carry the sensor A, and thus we allocate the task A to u_1 and u_3, and then the maximum workload of the u_1 and u_3 declines to 1 and 2, respectively. For task t_2, all users can complete it, but all users are competitive. In this case, we firstly select the users who carry fewer sensors to complete task; i.e., we select u_1 and u_2 to complete t_2. After that, the maximum workload of u_1 and u_2 declines to 0 and there are no more tasks that can be completed because of the limited capacity of all users. Thus, the number of completed tasks is 2 after particle initialization. The allocation result is summarized in Table 3.

4.5. Particle Updating. In general, the greedy strategy in the particle initialization cannot guarantee obtaining global

Table 1: Task set.

Tasks	Sensor Required	p
t_1	A	2
t_2	B	2
t_3	C	2
t_4	D	2

Table 2: User set.

Users	Sensors Preloaded	Maximum Workload
u_1	A, B, C	2
u_2	B, D	1
u_3	A, B, C, D	3

Table 3: Task allocation.

Tasks	Users Selected
t_1	u_1, u_3
t_2	u_1, u_2
t_3	u_3
t_4	u_3

optimal solution because this strategy selects the best local participants for every task. Thus, several refined strategies must be designed to improve this situation. Inspired by our previous work [22], we incorporate the mutation operation and crossover operation in GA to update the particles.

Formula (2) defines three parts for velocity updating. We incorporate the mutation operation in the first part. Tasks coordinate participants through the mutation operation. The updating formula is defined as follows:

$$A_i^t = w \otimes M_u\left(X_i^{t-1}\right) = \begin{cases} M_u\left(X_i^{t-1}\right) & r_1 < w \\ X_i^{t-1} & \text{esle}, \end{cases} \tag{5}$$

where M_u is the mutation operation with the probability of w. r_1 is random number between 0 and 1. The mutation operation randomly selects two points of a particle and then changes the value of all points between the two points. Note that the mutation operation must ensure satisfying all constraints in formula (1). The operation is illustrated in Figure 2.

The inertia weight influences the convergence and search ability. When we set a smaller w, PSO performs with high ability in local search; otherwise it performs with high ability in global search. In the early period of the algorithm, we must pay more attention to the diversity and global search ability of particles. To guarantee the convergence of algorithm, we must focus on the local search ability in the late period. Therefore, w should be reduced with the iterations. The updated function is expressed as follows:

$$w = w_{\max} - \text{iters}_{\text{cur}} \times \frac{w_{\max} - w_{\min}}{\text{iters}_{\max}}, \tag{6}$$

where $w_{\max}$ and $w_{\min}$ represent the maximum and minimum value of w, respectively. $\text{iters}_{\text{cur}}$ is the current iteration and $\text{iters}_{\max}$ is the maximum iterations.

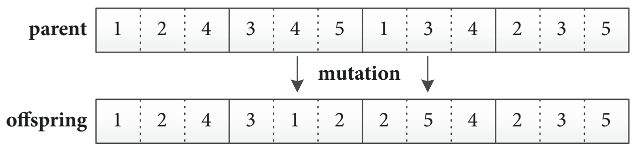


Figure 2: Mutation operation.

The second and third parts of formula (2) are the cognition of particle, indicating a particle learns from its previous optimal position and global optimal position. For these parts, we adopt the notion of crossover operation in the GA to update the particles, and the updating formulas are defined as follows:

$$\begin{aligned} B_i^t &= c_1 \oplus C_p\left(A_i^t, pBest^{t-1}\right) \\ &= \begin{cases} C_p\left(A_i^t, pBest^{t-1}\right) & r_2 < c_1 \\ A_i^t & \text{else} \end{cases} \\ X_i^t &= c_2 \oplus C_g\left(B_i^t, gBest^{t-1}\right) \\ &= \begin{cases} C_g\left(B_i^t, gBest^{t-1}\right) & r_3 < c_2 \\ B_i^t & \text{else}, \end{cases} \end{aligned} \tag{7}$$

where c_1 and c_2 are the crossover probabilities. Particles learn from previous optimal and current global optimal positions through crossover operation and finally converge to the optimal solution. The crossover operation must also ensure satisfying all constraints in formula (1). The operation is depicted in Figure 3.

In PSO, c_1 and c_2 are important parameters that reflect information exchange among particles. If we set greater c_1, particles intend to learn more from their personal scheme and hover within a local range. If we set greater c_2, particle may get into local optimality early. To keep the diversity of population, particles must learn more from their personal optimal scheme with great probability during the early stage of iteration. However, to guarantee the convergence of the algorithm, participates must learn more from global best scheme with great probability in the late stage of iterations. Thus c_1 should decrease while c_2 increase with the iterations. We update two acceleration factors with the linear strategy. The updated formulas are expressed as follows:

$$\begin{aligned} c_1 &= c_1\text{\_start} - \frac{c_1\text{\_start} - c_1\text{\_end}}{\text{iters}_{\max}} \times \text{iters}_{\text{cur}} \\ c_2 &= c_2\text{\_start} - \frac{c_2\text{\_start} - c_2\text{\_end}}{\text{iters}_{\max}} \times \text{iters}_{\text{cur}}, \end{aligned} \tag{8}$$

where c_1_start and c_2_start are the initial values of c_1 and c_2, respectively. c_1_end and c_2_end are the final values of c_1 and c_2, respectively.

In summary, the position of the particle i at iteration t can be updated according to following formula:

```
Input: The result of Algorithm 1
Output: Number of completed tasks
(1) Initialize the relative parameters;
(2) Iteration = 1;
(3) Get the particle $i$ by operation of mutation and crossover;
(4) Compute the number of completed tasks and update the $pbest_i$ and $gbest$ if necessary;
(5) Iteration++;
(6) If the value of $gbest$ keeps the same for twenty times, terminate the iteration;
(7) If iteration < Maximum iteration, go to (3);
(8) Output the number of completed tasks.
```

Algorithm 2: GDPSOGA.

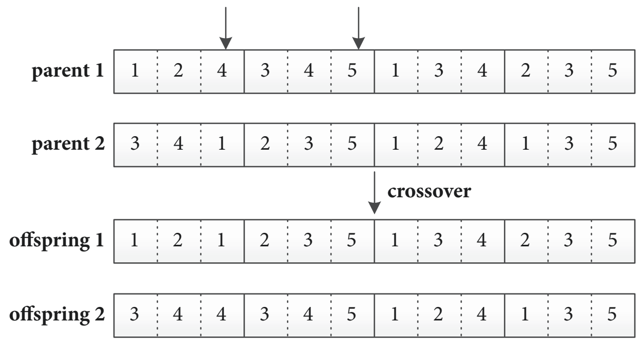


Figure 3: Crossover operation.

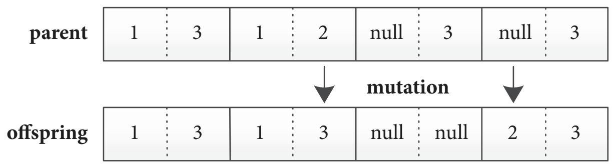


Figure 4: Mutation operation of tasks in Table 1.

$$X_i^t = c_2 \oplus C_g\left(c_1 \oplus C_p\left(w \otimes M_u\left(X_i^{t-1}\right), pBest_i^{t-1}\right), gBest^{t-1}\right) \quad (9)$$

4.6. Algorithm Overview. To address the MTHP problem, we propose the GDPSOGA. The pseudocode is presented in Algorithm 2. According to Algorithm 2, we initiate the particles with the result of Algorithm 1. To improve the result of greedy scheme, we incorporate the notions of mutation and crossover operations in GA into the DPSO. The integration can maintain not only the diversity of population, but also preferable characteristics of offspring population.

Taking Tables 1 and 2 as an example, there are two tasks that can be completed after particle initialization. If a particle changes the allocation of t_2 to u_1 and u_3, then, u_2 has spare capacity to complete t_4. So t_4 can be completed by u_2 and u_3. The mutation operation is presented in Figure 4. Thus, the number of completed tasks can be improved to 3. Note that because of the randomness of mutation, we only present a possible case in the mutation operation.

4.7. Time Complexity

Theorem 4. *The time complexity of GDPSOGA is $O(n \times k \log k + R \times S \times n \times p)$, where n is the number of tasks. p indicates the number of candidate participants that every task requires. k is the number of participants that carry the required sensor. R denotes the maximum number of iterations. S represents the population size of the GDPSOGA.*

Proof. In the first step of Algorithm 1, we sort the participants according to their moving time and sensors; the complexity is $O(n \times k \log k)$.

For the second step of Algorithm 1, we greedily select participants to initiate the particle. The length of particles is $n \times p$. Thus the time complexity of particle initialization is $O(S \times n \times p)$.

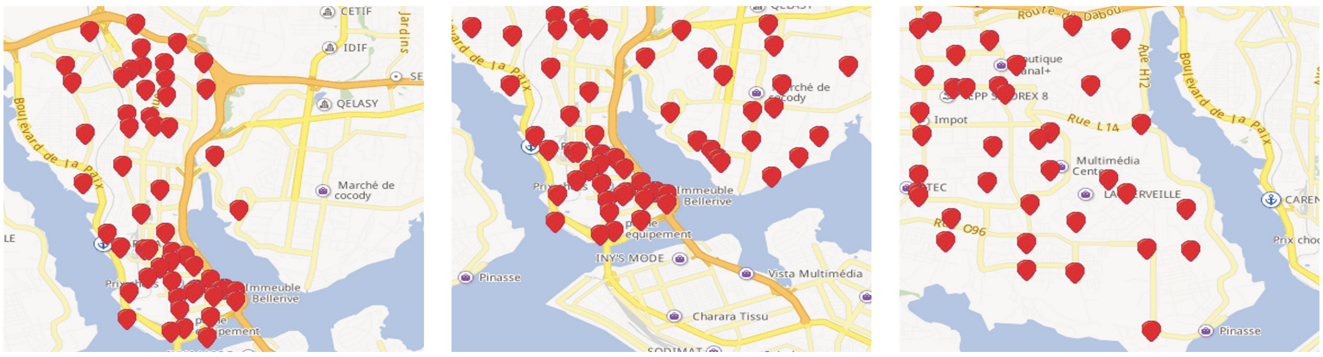
During each round of the iteration, the GDPSOGA updates particles by incorporating the two-point mutation and two-point crossover operations in the GA. In the worst case, the whole particle is updated. Thus the time complexity of updating is $O(S \times n \times p)$. Furthermore, the complexity of updating R iterations is $O(R \times S \times n \times p)$.

In summary, the time complexity of the GDPSOGA in the worst case is $O(n \times k \log k + R \times S \times n \times p)$. □

5. Evaluation and Discussion

In this section, we conduct experiments to examine the performance of our proposed algorithm under different settings. First, we present the dataset and experiment settings. Then, we introduce some baseline algorithms for evaluation. Finally, the detailed results of the proposed and baseline algorithms are presented and analyzed.

5.1. Dataset. We evaluate the performance of the proposed algorithm using the D4D dataset [23]. It is a large-scale real-world dataset that contains two types of data. One type is the location information of 1231 cell towers. The other type comprises individual call records for over 50,000 users of the Orange Group during two weeks in Ivory Coast. Each record includes user id, connection time, and cell tower. We design experiments based on the location information of cell towers and users.



(a) Compact (b) Hybrid (c) Scattered

Figure 5: Three types of tasks distribution.

Table 4: Relative parameters of the GDPSOGA.

Parameter	Value or Range
population size	100
inertia value w	0.9 to 0.4
acceleration factor c_1	0.9 to 0.2
acceleration factor c_2	0.4 to 0.9
maximum iteration	1000

5.2. Experiment Settings. For candidate participants, we select the candidate participants who made a phone call at the scope of cell towers between 18:00 and 19:00 on November 11, 2011. The maximum workload of users is set randomly between 5 and 10 and the walking speed of users is randomly set between 65 and 70 meters per minute. For tasks, we set the location of cell towers as the location of tasks. The sensor type that tasks require is randomly set between "A" and "E". The duration of tasks is randomly set to 1 to 3 minutes. Furthermore, we select different task distributions to measure the performance of our proposed method.

(i) Compact Distribution. Tasks are distributed compactly in a special region, and the distance among the tasks is relatively close.

(ii) Scattered Distribution. Tasks are scattered over the target region, and the distance among the tasks is relatively far.

(iii) Hybrid Distribution. Tasks are randomly distributed in the target area.

An illustration of three types of distribution is presented in Figure 5.

For the GDPSOGA, the relative parameters are listed in Table 4.

5.3. Baseline Algorithms. To compare the effectiveness of the proposed algorithm, we design three baseline task allocation methods as follows.

(i) Random Allocation (RA). This method selects participants randomly provided that the participants satisfy all constraints in formula (1) and without considering any heuristic strategy in this scheme. We repeat the random selection for 30 times and compute the average value of completed tasks.

(ii) Greedy Strategy Based Allocation (GSA). This method selects participants as presented in Algorithm 1, which considers the moving time, maximum workload of participants, and sensor number of devices, and does not implement the process of PSO.

(iii) Discrete PSO with GA (DPSOGA). This method selects participants using the PSO with mutation and crossover operations, without any heuristic strategy. Similar to the GDPSOGA, the DPSOGA terminates the iteration after the global optimal solution is kept unchanged for 20 times.

5.4. Number of Completed Tasks Comparison. We complete experiments in different situations to evaluate the performance of GDPSOGA. In the first set of simulations, we evaluate the influence of the number of participants required for every task. We fix the number of tasks at 300 and total number of candidate participants at 200. We vary p from 4 to 12. Figure 6 demonstrates that the number of completed tasks decreases when p increases. This is reasonable because finding a sufficient number of participants to complete task is difficult given the limited candidate participants. GSA and GDPSOGA can complete most of tasks when p is set to 4. With the increment of p, GSA only selects the optimal participants for every task under limited participant resource. GDPSOGA coordinates the user resource after GSA and thus obtains better results than GSA.

In the next set of simulations, we evaluate the influence of number of tasks. Here we set number of candidate participants at 160 and the required number of participants by every task at 5. The experiments are completed with varying number of tasks from 200 to 300. Figure 7 displays the results. In general, the number of completed tasks increases when the task number increases. However, RA and GSA cannot improve the result when the number of tasks reaches a certain extent because, on one hand, the number of candidate participants is limited. On the other hand, the two schemes merely select participants for every task and do not coordinate all candidate participants for all tasks.

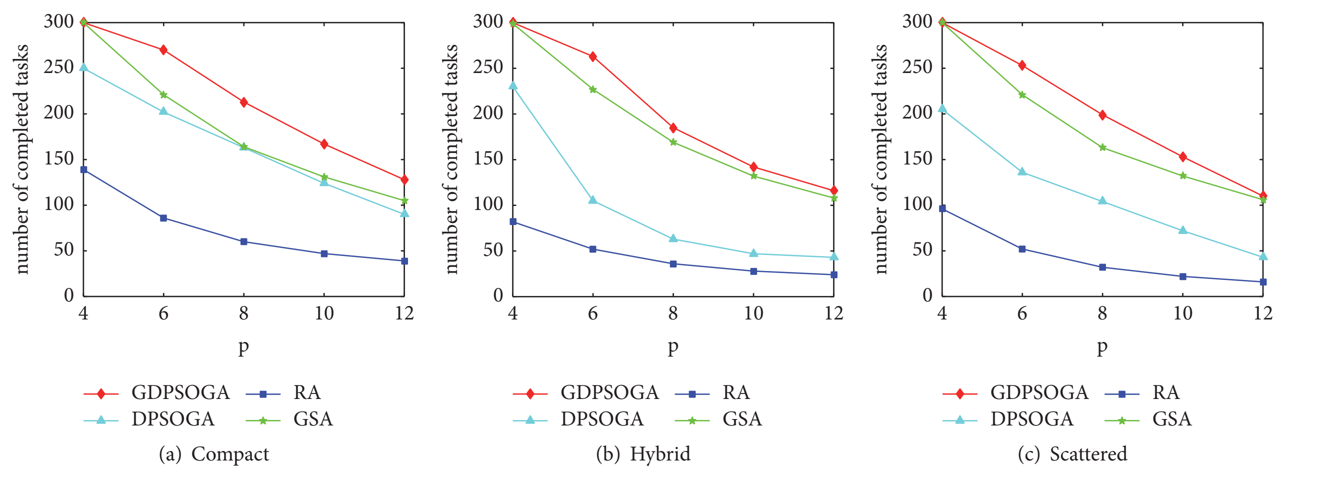


Figure 6: Performance comparison under different p.

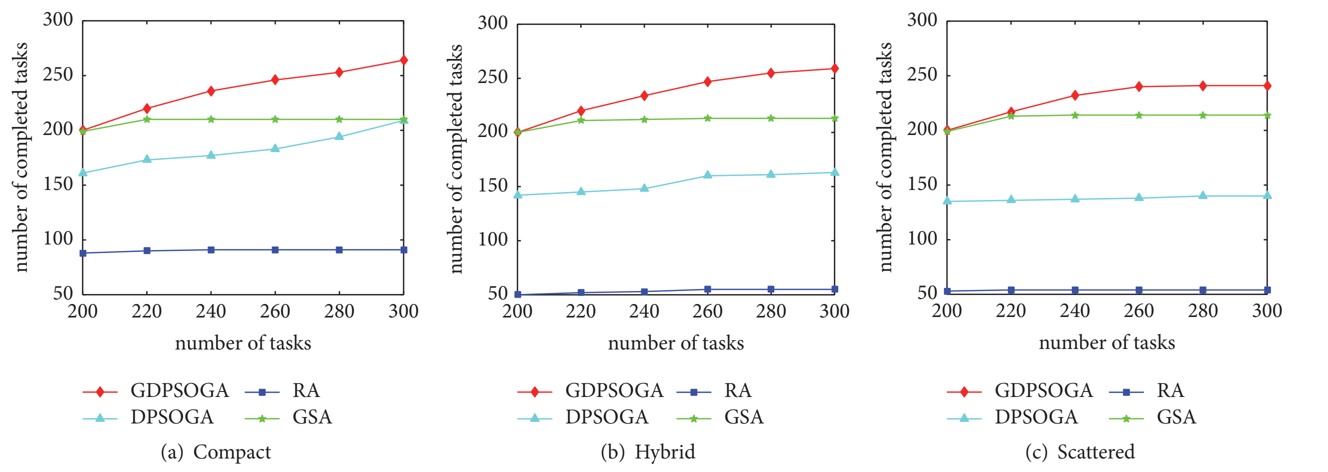


Figure 7: Performance comparison under different number of tasks.

In the last set of simulations, we fix the number of tasks at 300 and the required number of participants by every task at 5. Here, we vary the number of candidate participants from 100 to 200. Figure 8 illustrates the results under different task distributions. The number of completed tasks increases with candidate participants because additional candidate participants may offer increased possible optimized allocation schemes.

In general, the completion ratio of tasks is higher in compact distribution than in the two other distributions. For example, the average completion ratio of the three types of distribution in the second simulation is 95.2%, 94.9%, and 92.3%. Participants may not need to travel long distance to complete tasks because the locations among tasks are nearest. Thus, more tasks can be completed within the tasks during time.

Among the different task assignment schemes, the GDPSOGA outperforms other schemes because the GDPSOGA combines the advantage of the DPSO and GSA. Specifically, a heuristic strategy, which can effectively reduce the solution space of our problem, is adopted in the early stage of the GDPSOGA to obtain the optimal or suboptimal solution. However, the heuristic strategy selects the local optimal solution for every task. It cannot guarantee the optimal solution for all tasks. Thus several strategies must be adopted to coordinate the participants to complete as many tasks as possible. The GDPSOGA adopts the crossover and mutation operations to match the participants among tasks after a greedy strategy participant selection; thereby it improves the results.

5.5. Computing Time. In this section, we compare the computing time of all algorithms. We employ Java language to implement the algorithm. All the experiments are conducted on a PC with 3.10 GHz CPU and 16 GB memory.

Table 5 shows a comparison of computing time of the first simulation in the compact distribution. From Table 5, we can see that RA takes the least time because it just selects participants as long as participants satisfy all the constraints and does not consider any heuristic strategy. GSA firstly sorts the participants according to the moving time, maximum workload, and the sensors of participants and then greedily selects participants according to the sort results. So it takes more time than the RA. Our algorithm takes longer time than

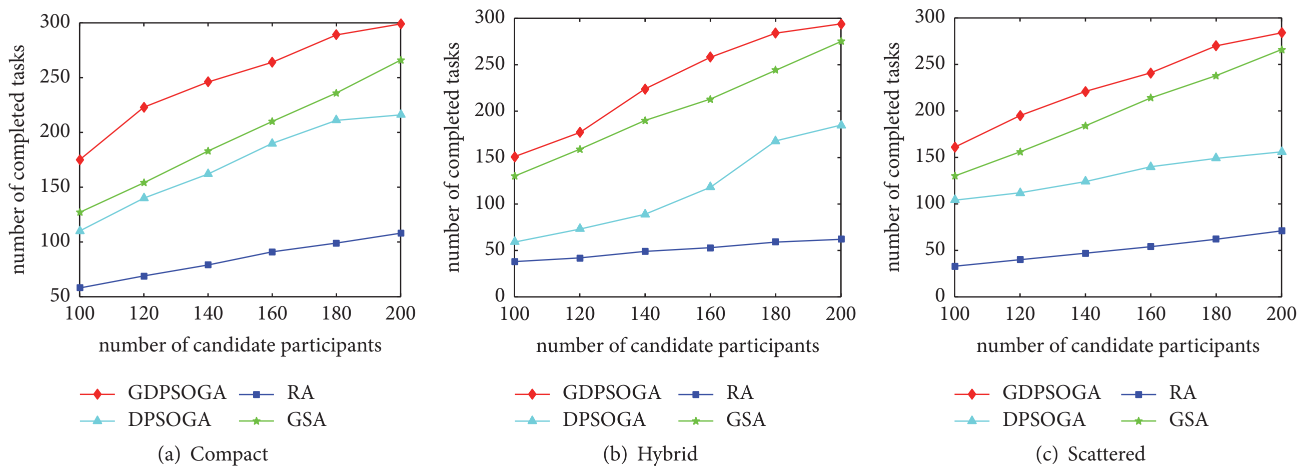


Figure 8: Performance comparison under different number of candidate participants.

Table 5: Comparison of CPU time (s) of second simulation in compact distribution.

Algorithms	p				
	4	6	8	10	12
GDPSOGA	1.294	1.816	2.238	2.452	2.76
DPSOGA	1.124	1.135	1.654	2.229	2.578
GSA	0.102	0.105	0.11	0.115	0.124
RA	0.02	0.023	0.027	0.028	0.032

GSA due to the iteration of PSO to coordinate the participant resource among tasks. But it is worthy, because it improves the number of completed tasks and obtains the results within reasonable time. Note that without any heuristic strategy DPSOGA may fall into local optimum, so it takes less time than GDPSOGA.

6. Conclusion

In this study, we focus on the problem of MTHP in MCS to maximize the completed tasks while satisfying certain constraints. In contrast to other existing works, this study considers the heterogeneity among participants. We propose the GDPSOGA to solve the problem, which incorporates the random two-point crossover and random two-point mutation operations in GA into DPSO. Extensive simulations conducted over a real-life dataset confirm the efficiency of the proposed algorithm. However, in this study, we assume that the tasks are static and no new sensing tasks emerge after the task allocation starts. In our future work, we will explore the allocation schemes for dynamic tasks.

Data Availability

Dataset used in this article is released by D4D Challenge. Participants in this challenge signed an agreement on data confidentiality. So it is available in limited way. One of the coauthors participated in this challenge. So access to the dataset can be got. Some introduction about the dataset can be obtained from https://arxiv.org/abs/1210.0137.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (nos. U1705262, 61672159, and 61772136), the Outstanding Youth Rolling Project of Fujian Province under Grant no. 2018J07005, the Technology Innovation Platform Project of Fujian Province under Grant nos. 2014H2005 and 2009J1007, the Fujian Collaborative Innovation Center for Big Data Application in Governments, and the Fujian Engineering Research Center of Big Data Analysis and Processing.

References

[1] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson, “Cooperative transit tracking using smart-phones,” in *Proceedings of the 8th ACM International Conference on Embedded Networked Sensor Systems, SenSys 2010*, pp. 85–98, November 2010.

[2] N. Pham, R. K. Ganti, Y. S. Uddin, S. Nath, and T. Abdelzaher, “Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 5970, pp. 114–130, 2010.

[3] Y. Zheng, F. Liu, and H.-P. Hsieh, “U-air: when urban air quality inference meets big data,” in *Proceedings of the 19th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13), pp. 1436–1444, Chicago, Ill, USA, August 2013.

[4] M. Demirbas, M. A. Bayir, C. G. Akcora, Y. S. Yilmaz, and H. Ferhatosmanoglu, "Crowd-sourced sensing and collaboration using twitter," in *Proceedings of the 2010 IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks", WoWMoM 2010*, can, June 2010.

[5] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, "A PSO-Optimized Real-Time Fault-Tolerant Task Allocation Algorithm in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3236–3249, 2015.

[6] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing*, vol. 6030 of *Lecture Notes in Computer Science*, pp. 138–155, Springer, Berlin, Germany, 2010.

[7] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using context annotated mobility profiles to recruit data collectors in participatory sensing," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 5561, pp. 52–69, 2009.

[8] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes, "ICrowd: Near-Optimal Task Allocation for Piggyback Crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010–2022, 2016.

[9] M. Zhang, P. Yang, C. Tian et al., "Quality-aware sensing coverage in budget-constrained mobile crowdsensing networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7698–7707, 2016.

[10] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proceedings of the 34th IEEE Annual Conference on Computer Communications and Networks, IEEE INFOCOM 2015*, pp. 2254–2262, May 2015.

[11] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 703–714, 2014.

[12] L. Wang, D. Zhang, A. Pathak et al., "CCS-TA: quality-guaranteed online task allocation in compressive crowdsensing," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*, pp. 683–694, ACM, Osaka, Japan, September 2015.

[13] Z. Yu, J. Zhou, W. Guo, L. Guo, and Z. Yu, "Participant selection for t-sweep k-coverage crowd sensing tasks," *World Wide Web*, pp. 1–18, 2017.

[14] H. Li, T. Li, and Y. Wang, "Dynamic participant recruitment of mobile crowd sensing for heterogeneous sensing tasks," in *Proceedings of the 12th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2015*, pp. 136–144, Dallas, TX, USA, October 2015.

[15] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, and D. Zhang, "TaskMe: Multi-task allocation in Mobile Crowd Sensing," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2016*, pp. 403–414, deu, September 2016.

[16] J. T. Wang, Y. S. Wang, D. Q. Zhang, F. Wang, Y. D. He, and L. T. Ma, "PSAllocator: multi-task allocation for participatory sensing with sensing capability constraints," in *Proceedings of the International Conference on Computer-Supported Cooperative Work and Social Computing (CSCW)*, pp. 1139–1151, 2015.

[17] Z. Song, C. H. Liu, J. Wu, J. Ma, and W. Wang, "QoI-aware multitask-oriented dynamic participant selection with budget constraints," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4618–4632, 2014.

[18] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "ActiveCrowd: A Framework for Optimized Multitask Allocation in Mobile Crowdsensing Systems," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 392–403, 2017.

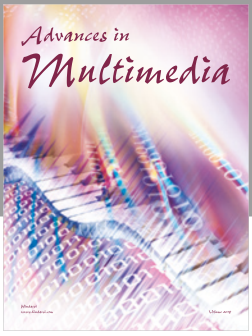
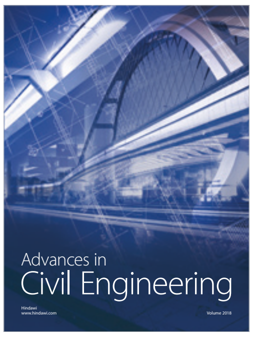
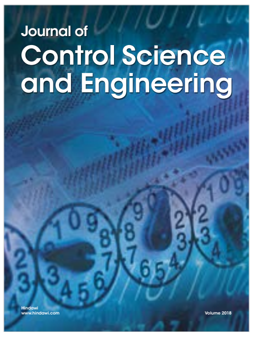
[19] S. He, D. Shin, J. Zhang, and J. Chen, "Near-optimal allocation algorithms for location-dependent tasks in crowdsensing," *IEEE Transactions on Vehicular Technology*, pp. 1-1, 2017.

[20] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. S. Shen, "Providing Task Allocation and Secure Deduplication for Mobile Crowdsensing via Fog Computing," *IEEE Transactions on Dependable and Secure Computing*, pp. 1-1.

[21] D.-J. Chang, A. H. Desoky, M. Ouyang, and E. C. Rouchka, "Compute pairwise Manhattan distance and Pearson correlation coefficient of data points with GPU," in *Proceedings of the 10th ACIS Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2009, In conjunction with IWEA 2009 and WEACR 2009*, pp. 501–506, Daegu, South Korea, May 2009.

[22] W. Guo, W. Hong, B. Zhang, Y. Chen, and N. Xiong, "Reliable adaptive data aggregation route strategy for a trade-off between energy and lifetime in WSNs," *Sensors*, vol. 14, no. 9, pp. 16972–16993, 2014.

[23] V. D. Blondel, M. Esch, C. Chan et al., "Data for Development: the D4D Challenge on Mobile Phone Data," 2012, https://arxiv.org/abs/1210.0137.

Hindawi

Submit your manuscripts at
www.hindawi.com