




Maximizing user type diversity for task assignment in crowdsourcing

Ana Wang^{1,2,3} · Meirui Ren³ · Hailong Ma³ · Lichen Zhang^{1,2,3} · Peng Li^{1,2,3} · Longjiang Guo^{1,2,3} 

Published online: 3 October 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Crowdsourcing employs numerous users to perform certain tasks, in which task assignment is a challenging issue. Existing researches on task assignment mainly consider spatial–temporal diversity and capacity diversity, but not focus on the type diversity of users, which may lead to low quality of tasks. This paper formalizes a novel task assignment problem in crowdsourcing, where a task needs the cooperation of various types of users, and the quality of a task is highly related to the various types of the recruited users. Therefore, the goal of the problem is to maximize the user type diversity subject to limited task budget. This paper uses three heuristic algorithms to try to resolve this problem, so as to maximize user type diversity. Through extensive evaluation, the proposed algorithm Unit Reward-based Greedy Algorithm by Type obviously improves the user type diversity under different user type distributions.

Ana Wang and Meirui Ren have contributed equally to this work.

✉ Lichen Zhang
zhanglichen@snnu.edu.cn

✉ Longjiang Guo
longjiangguo@snnu.edu.cn

Ana Wang
ana_wang@snnu.edu.cn

Meirui Ren
meirui ren@snnu.edu.cn

Hailong Ma
mahailong@snnu.edu.cn

Peng Li
lipeng@snnu.edu.cn

¹ Key Laboratory of Modern Teaching Technology, Ministry of Education, Xi'an 710062, China

² Engineering Laboratory of Teaching Information Technology of Shaanxi Province, Xi'an 710119, China

³ School of Computer Science, Shaanxi Normal University, Xi'an 710119, China

Keywords Crowdsourcing · Crowdsensing · Task assignment · User type diversity

1 Introduction

Crowdsourcing (Xing et al. 2019; Qiao et al. 2018; Li et al. 2018c) mostly comprises taking a large task and separating it into many smaller tasks on which a crowd of people can work separately. It involves obtaining work, information, or data from a large group of people who present their data via social communication software and smart phone apps on the Internet. In the existing research literature, crowdsourcing mainly has the following research directions, namely task assignment (Cai et al. 2020; Wang and Wang 2016; Tong et al. 2019), incentive mechanism (Wang et al. 2020) and privacy protection (Liu et al. 2020).

This paper focuses on how to maximize user type diversity. For example, mobile crowdsensing (MCS) (Ganti et al. 2011; Wei et al. 2018; Zhu et al. 2019; Duan et al. 2019) is a kind of crowdsourcing. It is a new paradigm and widely applied in many areas, which mainly utilizes the sensing, computing, storage and communication functions of mobile smart devices. It regards mobile smart devices carried by mobile users as wireless sensors with powerful functions and completes the sensing task together through their cooperation. In MCS, a sensing task is assigned to amounts of normal users, who use their carried smart devices to sense data dynamically without interference of users (Li et al. 2016, 2018a; Wang et al. 2018a). As we know, a sensing task usually needs the cooperation of numerous users to sense data. In daily life, users may have different roles or types, such as worker, teacher, student, etc. The sensed data obtained from a user with a specific type may be quite different from users with other types. Therefore, it is beneficial to improve the quality of a task if assigning the task to all kinds of types of users more uniformly. In such tasks, it is expected to maximizing the user type diversity.

It is very important to study task allocation with the consideration of the diversity of user types. There are many researches on task allocation in MCS applications, mostly of which are related with different objective functions and task constraints. The constraints of the perceptual tasks in the existing research are all related to the quality, budget, time and position constraints of the task. The objective function is around maximizing profit, minimizing cost, maximizing coverage quality, etc., but does not consider the user type diversity as the goal function for research. For some specific perception tasks, the quality of a single type of user's completion is lower, but if multiple different types of users collect the perception data, it will be more convincing. For example, users take pictures to record water quality in different places. This task requires multiple types of users to complete, because the pixels of pictures taken by different types of smart devices are different, and the data collected by a single type of smart device is not enough to explain the quality of water quality. Therefore, it is of great significance to study task allocation with the consideration of the diversity of user types.

It is shown that the user type diversity has great impact on the quality of task in crowdsourcing. An interesting example for user type diversity in crowdsourcing is to measure pollution in cities. In the task, we can record the pollution in different places

by using smart devices to take pictures. This task needs a large number of users to complete together. Since the system of each user's smart device is different, in order to ensure the accuracy of the measured data, we should choose multiple types of devices to collect pollution information. Therefore, it is shown that the user type diversity has great impact on the quality of task in crowdsourcing. In such tasks, we also expect to maximizing the user type diversity. Another practical example for user type diversity is in the mobile learning. In mobile learning environment, learners' learning situation is very important (Yao 2017). Mobile learning system recommends learning contents suitable for mobile learners according to their learning situations (Alhamid et al. 2016). Mobile learning system needs to collect learners' sensing data of learning situation. In practical applications, learners have various types of hand-held intelligent devices. When learners with different devices sense the same learning situation, the sensing data is inconsistent. Mobile learning system needs different sensing data which sensed by different types of device when establishing recommendation model. This requires maximizing the diversity of user types. The third practical example for user type diversity is in the online education. Learning teaming skills is an important part of online education. And in larger classes, specifically MOOC platform (Liao et al. 2019) courses or mobile learning on campus via opportunistic social networks (Kong et al. 2019) which could potentially involve hundreds of teams. In each team, collaborative learning is often needed (Yuan and Cao 2019). For the best learning outcome, the diversity of student(user) types is often needed. On such online education platform, teams should form eclectically as tasks. In such tasks, it is expected to maximizing the user type diversity.

However, the current researches mainly focuses on spatial-temporal diversity (Cheng et al. 2015a; Cranshaw et al. 2010; Kazemi and Shahabi 2012) and capability diversity (Cohen and Yashinski 2017; Cheng et al. 2015b; Wang et al. 2018b; Li et al. 2019), without considering the user type diversity. Furthermore, the existing task assignment mechanisms mainly focus on the optimization of some goals, such as minimizing cost (Yu et al. 2016; Wang et al. 2018c; Zhang et al. 2018; Duan et al. 2017), maximizing profit (Li et al. 2018b), maximizing social welfare (Wang et al. 2016, 2019), maximizing user coverage quality (Abououf et al. 2019; Zhang et al. 2016; Yin et al. 2017; Gong et al. 2019; Tao and Song 2019), and so on. Cheng et al. (2015b) consider multi-skill spatial crowdsourcing and maximize workers benefit under the budget constraint. Wang et al. (2018b) study the location-aware and location diversity based on dynamic crowdsensing system, where workers move over time and tasks arrive stochastically. Li et al. (2019) focus on the room allocation problem with capacity diversity and budget constraints. The above-related works do not take into account the type diversity, which may be lead to a low quality.

Motivated by the above observation, this paper formalizes a novel task assignment problem in MCS, where a task needs the cooperation of various types of users, and the quality of a task is highly related to the types of the recruited users. The main contributions are given as follows.

1. The paper proposes a novel type diversity-oriented task assignment problem under the constraints of budget, the latest ending time, the required types and the required

- number of users, in which the concept of entropy is introduced to measure the optimization goal of the problem.
2. In order to maximize the type diversity of the selected users and meet the type requirements of the task, this paper proposes three heuristic algorithms: Cost-based Greedy Algorithm (C-GA), Cost-based Greedy Algorithm by Type (C-GAT), and Unit Reward-based Greedy Algorithm by Type (UR-GAT).
 3. This paper conducts extensive simulations to show that Unit Reward-based Greedy Algorithm by Type has better user type diversity and total profit than the other algorithms under different user type distributions.

2 Model and problem formulation

In this section, a task assignment model that maximizes the type diversity of crowd users in crowdsourcing is described. Then, a formalized representation of the model is given under the assumption that all crowd users and related task information are known. For easy of reference, the important symbols and their meanings used are listed in Table 1.

2.1 Model

A task $\tau = \langle d_\tau, b_\tau, n_\tau, T_\tau \rangle$ is represented by quadruples whose four elements are respectively the latest end time of the task τ , the budget of τ , the number of users required by τ and the set of types required by τ , where the element $T_\tau \subseteq \{1, 2, \dots, k\}$ and $|T_\tau| > 1$. In our model, any user in the crowd of users $U = \{u_1, \dots, u_i, \dots, u_m\}$ is interested in participating in crowdsourcing task. The i th user is represented as $u_i = \langle c_i, r_i, s_i, t_i \rangle$, in which four elements represent respectively the type, the latest start time the cost and the reward for participating crowdsourcing task; that is, each user belongs to a type and the user u_i is unavailable before the time s_i . We also assume that a task always requires many users with distinct user types to collaborate with each other. Obviously, different type users have their own advantages, so the above assumption is consistent with many real-world scenarios. $\{t_{i_1}, t_{i_2}, \dots, t_{i_{n_\tau}}\} (t_{i_j} \in T_\tau, j = 1, 2, \dots, n_\tau)$ is the statistical set of user types participating in task τ , where n_τ is the number of users required by the task τ . Based on statistical analysis on the data set, we can obtain a distribution of user types required by the task τ , which is an important factor affecting the diversity of user types.

2.2 Problem formulation

Based on the above model, if we want to improve the quality of task completion, the type distribution of users participating in a task should be as uniform as possible; i.e., we should increase the type diversity of users as much as possible. Motivated by the analysis, in order to maximize the type diversity of users within the given task budget, we use the concept of information entropy to measure the type diversity of users participating in task and formalize the task allocation problem as follows:

Table 1 Definition of variable symbols

Notation	Description
U, T	The set of users and type sets
τ	Task τ
u_i	The i th user
c_i	The cost of u_i
r_i	The reward of u_i
s_i	The latest start time of u_i
t_i	The type of u_i
d_τ	The latest end time of the task τ
T_τ	The set of types required by the task τ
n_τ	The number of users required by the task τ
b_τ	The budget of task τ
I_i^j	Indicator variable I_i^j shows Whether the user u_i belongs to the type j
I_τ^j	Indicator variable I_τ^j shows Whether type j is required by the task τ
k	The number of types
m	The number of users
x_i	Indicator variable x_i shows whether user u_i is assigned to task τ
U_j	The set of users with type j in ascending order based on cost
$count_j$	The number of users required with type j
p_j	The ratio of $count_j$ to n_τ

$$\begin{aligned}
 &\text{Max : } - \sum_{j \in T} p_j \cdot \log(p_j) \\
 &s.t. 1) \sum_{i=1}^m r_i \cdot x_i \leq b_\tau; \\
 &2) s_i \cdot x_i \leq d_\tau, \forall i \in \{1, \dots, m\}; \\
 &3) \sum_{i=1}^m x_i \leq n_\tau; \\
 &4) \sum_{i=1}^m x_i \cdot I_i^j \geq I_\tau^j, \forall j \in \{1, \dots, k\}; \\
 &5) p_j = count_j / n_\tau; \quad count_j = \sum_{i \in U_j} x_i; \quad x_i \in \{0, 1\}, \forall i \in \{1, \dots, m\};
 \end{aligned} \tag{1}$$

In Eq. 1, the optimization objective is to maximize the information entropy $-\sum_{j \in T} p_j \cdot \log(p_j)$, where $p_j = \text{count}_j / n_\tau$ is the ratio of the number of users with type j to the total number of users required by task τ (see C-GAT algorithm for details).

Constraint 1 ensures that the total rewards of the users assigned to task τ does not exceed the budget of task τ . Constraint 2 means that the latest start time of each user assigned to task τ cannot exceed the latest end time of task τ . Constraint 3 implies that the number of users assigned to task τ should not exceed the total number of users required by task τ . Constraint 4 indicates that if the task τ requires the users with type j (i.e., $I_\tau^j = 1$), at least one user with type j is assigned to the task. Here, $I_i^j = 1$ means that user u_i belongs to the type type j . In Constraint 5, if the user u_i is assigned to the task τ , the indicator variable $x_i = 1$. Therefore, $\text{count}_j = \sum_{i \in U_j} x_i$ represents the total number of users with type j assigned to task τ .

The mathematical model of above mentioned problem is formulated as a 0–1 type integer programming problem (Li et al. 2018a, b), so the problem is also NP-hard.

3 The proposed task assignment algorithms

For solving the proposed problem efficiently, this paper proposes three heuristic algorithms, named C-GA (Cost-based Greedy Algorithm), C-GAT (Cost-based Greedy Algorithm by Type), and UR-GAT (Unit Reward-based Greedy Algorithm by Type), respectively.

3.1 Cost-based Greedy Algorithm (C-GA)

Assume that the task assignment system contains one task, 6 types and 10 crowd users, where the user set is expressed as $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}\}$. The parameter values of tasks are as follows: the budget b_τ is 115, the required number of users n_τ is 4, the latest end time of the task d_τ is 23, the set of required user type is $T_\tau = \{1, 0, 1, 1, 0, 0\}$, and the parameters of users are shown in the Table 2.

First, select a set of candidate users. All users are sorted by cost in ascending order, and the sorted set is recorded as $\{u_9, u_4, u_2, u_7, u_5, u_1, u_8, u_6, u_{10}, u_3\}$.

Then, determine whether the user satisfies constraints successively from the candidate user set, that is: determine whether the latest start time of the user is less than the latest end time of the task and whether the user belongs to the type required by the task; whether the reward paid to the user is less than the budget of the task, and whether the number of users required for the task is not less than 0.

Because the latest start time of user u_{10} is greater than the latest end time of the task, the types that user u_5, u_7 and u_{10} belongs to are not the type required by the task, so the set of candidate users is $\{u_9, u_2, u_1, u_8, u_6, u_3\}$.

Select the user u_9 with the minimum cost from the candidate users set $\{u_9, u_2, u_1, u_8, u_6, u_3\}$, whose reward is less than the task's budget, and the number of users required by the task is 4, so the user is assigned to the task, Then update the budget of task as 96 and the number of users required by the task as 3. Select the next

Table 2 The parameters values of user

Number	Start time	Cost	Reward	Type
u_1	16	8	16	{1, 0, 0, 0, 0, 0}
u_2	23	8	16	{0, 0, 0, 1, 0, 0}
u_3	18	10	15	{0, 0, 1, 0, 0, 0}
u_4	19	5	19	{0, 1, 0, 0, 0, 0}
u_5	22	7	17	{0, 1, 0, 0, 0, 0}
u_6	20	9	15	{0, 0, 0, 1, 0, 0}
u_7	15	6	18	{0, 0, 0, 0, 0, 1}
u_8	17	8	17	{0, 0, 1, 0, 0, 0}
u_9	21	4	19	{1, 0, 0, 0, 0, 0}
u_{10}	24	9	20	{0, 0, 0, 0, 1, 0}

user u_2 in the set. Because this user meets constraints, the user u_2 is assigned to the task, with an updated task budget of 78 and the number of users required by the task is 2. Select the next user u_1 in the set. Because this user meets constraints, the user u_1 is assigned to the task, with a budget of 62 for the updated task and the number of users required by the task is 1. Select the next user u_8 in the set. Because this user meets constraints, the user u_8 is assigned to the task with a budget of 45 for the updated task and a number of 0 for the task. Select the next user u_6 in the set, because the number of users required by the task is equal to 0, so the constraint condition is not satisfied, and task assignment is finished.

Finally, the task assignment result set is returned as {1, 1, 0, 0, 0, 0, 0, 1, 1, 0}. That is, the task is assigned to u_1 , u_2 , u_8 and u_9 .

As shown in Algorithm 1, C-GA is depicted as follows. We sort users by cost in ascending order (line 1). If user u_i satisfies the required types of task τ while considering the latest ending time and the budget constraints, the task τ will be assigned to user u_i , $x_i = 1$ (line 2–5). Next, we update the budget and the required number of users, if the required number of task is not more than zero, then the loop will be broken (line 6–13). Finally, the algorithm returns the result of task assignment (line 14).

3.2 Cost-based Greedy Algorithm by Type (C-GAT)

In order to further understand the algorithm, C-GA algorithm is used to give the input data of the instance. The detailed instance task assignment process is described as follows.

First, filter out the users who do not meet the time constraints, count the users in each type to the corresponding set and sort them in ascending order of cost in the set. Because the latest start time of user u_{10} is greater than the latest end time of the task, so the set of candidate users is $\{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9\}$. Therefore, the user set T_1 of type t_1 is $\{u_9, u_1\}$, the user set T_2 of type t_2 is $\{u_4, u_5\}$, the user set T_3 of type t_3 is $\{u_8, u_3\}$, the user set T_4 of type t_4 is $\{u_2, u_6\}$, the user set T_5 of type t_5 is \emptyset , the

Algorithm 1 Cost-based Greedy Algorithm (C-GA).**Require:** task τ , user set U .**Ensure:** the task assignment results: $\{x_1, x_2, \dots, x_i, \dots, x_m | u_i \in U\}$.

1: Sort users by cost in ascending order;

2: **for all** $u_i \in U$ **do**3: **if** $s_i < d_\tau$ and $t_i \in T_\tau$ **then**4: **if** $b_\tau \geq r_i$ and $n_\tau > 0$ **then**5: set $x_i \leftarrow 1$;6: set $b_\tau \leftarrow b_\tau - r_i$;7: set $n_\tau \leftarrow n_\tau - 1$;8: **if** $n_\tau \leq 0$ **then**

9: break;

10: **end if**11: **end if**12: **end if**13: **end for**

14:

15: **return** $\{x_1, x_2, \dots, x_i, \dots, x_m\}$.

user set T_6 of type t_6 is $\{u_7\}$. Since the type set required for the task is $\{1, 0, 1, 1, 0, 0\}$, we need to select the appropriate user from the three sets T_1 , T_3 , and T_4 in turn.

Then, in order to ensure that the users selected by the task are equally distributed in each type, selecting users from the three sets of T_1 , T_3 , and T_4 is a rotation system, that is, selecting a user from T_1 , then choosing a user from T_3 , selecting a user from T_4 again. Select circularly the user whose reward paid is less than the current budget of the task from the sets of T_1 , T_3 and T_4 .

The further operation of the example is shown below. User u_9 is selected from T_1 , since this user satisfies the two constraints of task budget and the number of users required, task is assigned to user u_9 , update task budget is 96, and task required number of users is 3. Similarly, the user u_8 that satisfies the constraints is selected from T_3 and the task budget and the number of users required are respectively updated to 79 and 2; the User u_2 that satisfies the constraints is selected from T_4 and the task budget and the number of users required are respectively updated to 61 and 1 then the User u_1 that satisfies the constraints is selected from T_1 and the task budget and the number of users required are respectively updated to 45 and 0 Next, because the number of users required is 0 no user in T_3 can meet constraint. The task assignment ends.

Finally, the result set of the task assignment is returned as $\{1, 1, 0, 0, 0, 0, 0, 1, 1, 0\}$. That is, the task is assigned to u_1 , u_2 , u_8 and u_9 .

Perceptive tasks have diverse demands on user types. In order to maximize the diversity of user types, the selected users should be distributed as much as possible in the set of types required by the task. The C-GA algorithm proposed in this paper only selects the user with the lowest cost and does not consider that the user's type may lead to a reduction in the quality of task completion. Therefore, for tasks that have specific requirements of user type, we need to consider both the cost of performing the task and the type of user required by the task when selecting users, so this paper proposes a new greedy algorithm: C-GAT.

In order to maximize the type diversity, we consider the relationship between the required types of task and the type of users. As shown in Algorithm 2, C-GAT is

depicted as follows. First of all, for all users in set U , if the latest start time of the user is less than the latest end time of the task, we sort users with type j ($j \leq k$) by cost in ascending order as T_j (line 1–7). Next, we select the lowest cost user u_i from T_j (line 8, 9). Then, if user u_i satisfies the required types of task τ while considering the current number of users required and the budget constraints, the task τ will be assigned to user u_i , $x_i = 1$. The algorithm goes to the next loop when the user does not meet any of the constraints (line 10–22). Finally, the algorithm returns the result of task assignment (line 23).

Algorithm 2 Cost-based Greedy Algorithm by Type (C-GAT).

Require: task τ , user set U , the number of types k .

Ensure: the task assignment results: $\{x_1, x_2, \dots, x_i, \dots, x_m | u_i \in U\}$.

```

1: for all  $u_i \in U$  do
2:   for all type  $j$  do
3:     if  $s_i < d_\tau$  and  $u_i$  belongs to type  $j$  then
4:       Insert  $u_i$  into  $T_j$  by cost in ascending order;
5:     end if
6:   end for
7: end for
8: while  $j$  do
9:   select the lowest cost user  $u_i$  from  $T_j$ ;
10:  if  $b_\tau \geq r_i$  and  $n_\tau > 0$  and  $t_i \in T_\tau$  then
11:    set  $x_i \leftarrow 1$ ;
12:    set  $b_\tau \leftarrow b_\tau - r_i$ ;
13:    set  $n_\tau \leftarrow n_\tau - 1$ ;
14:    if  $n_\tau \leq 0$  then
15:      break;
16:    end if
17:  end if
18:   $j \leftarrow j + 1$ ;
19:  if  $j == k + 1$  then
20:     $j \leftarrow 1$ ;
21:  end if
22: end while
23: return  $\{x_1, x_2, \dots, x_i, \dots, x_m\}$ .

```

3.3 Unit Reward-based Greedy Algorithm by Type (UR-GAT)

In order to further understand UR-GAT algorithm, the task input data of the example given in the C-GA algorithm is used. Since the UR-GAT algorithm selects the user with the highest unit reward among the user types required by the task in turn, unit reward needs to be added user attributes. Unit reward refers to the ratio of reward to cost, because the user's reward and cost are shown in the Table 4, so only need to calculate the unit reward of users in the user set $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}\}$, modify the user's attribute value as shown in the following Table 3.

The detailed task assignment process is as follows:

Table 3 The parameters values of user

Number	Start time	Cost	Reward	Unit reward	Type
u_1	16	8	16	2	{1, 0, 0, 0, 0, 0}
u_2	23	8	16	2	{0, 0, 0, 1, 0, 0}
u_3	18	10	15	1.5	{0, 0, 1, 0, 0, 0}
u_4	19	5	19	3.8	{0, 1, 0, 0, 0, 0}
u_5	22	7	17	2.4	{0, 1, 0, 0, 0, 0}
u_6	20	9	15	1.7	{0, 0, 0, 1, 0, 0}
u_7	15	6	18	3	{0, 0, 0, 0, 0, 1}
u_8	17	8	17	2.1	{0, 0, 1, 0, 0, 0}
u_9	21	4	19	4.75	{0, 0, 0, 0, 1, 0}
u_{10}	24	9	20	2.2	{0, 1, 0, 0, 0, 0}

First, filter out the users who do not meet the time constraints, count the users in each type to the corresponding set and sort them in descending order of unit reward in the set. Because the latest start time of user u_{10} is greater than the latest end time of the task, so the set of candidate users is $\{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9\}$. Therefore, the user set T_1 of type t_1 is $\{u_9, u_1\}$, the user set T_2 of type t_2 is $\{u_4, u_5\}$, the user set T_3 of type t_3 is $\{u_8, u_3\}$, the user set T_4 of type t_4 is $\{u_6, u_2\}$, the user set T_5 of type t_5 is \emptyset , the user set T_6 of type t_6 is $\{u_7\}$. Since the type set required for the task is $\{1, 0, 1, 1, 0, 0\}$, we need to select the appropriate user from the three sets T_1 , T_3 , and T_4 in turn.

Select circularly the user whose reward paid is less than the current budget of the task from the sets of T_1 , T_3 , and T_4 when the number of users required by the task is greater than 0.

The further operation of the example is shown below. User u_9 is selected from T_1 , since this user satisfies the two constraints of task budget and the number of users required, task is assigned to user u_9 , update task budget is 96, and the current number of users required is 3. Similarly, the user u_8 that satisfies the constraints is selected from T_3 and the task budget and the number of users required are respectively updated to 79 and 2; the User u_6 that satisfies the constraints is selected from T_4 and the task budget and the number of users required are respectively updated to 64 and 1 then the User u_1 that satisfies the constraints is selected from T_1 and the task budget and the number of users required are respectively updated to 48 and 0 Next, because the current number of users required is 0, the task assignment ends.

Finally, the result set of the task assignment is returned as $\{1, 0, 0, 0, 0, 1, 0, 1, 1, 0\}$. That is, the task is assigned to u_1, u_6, u_8 and u_9 .

In C-GAT algorithm, the type and cost of the users are considered. However, the user has another attribute reward, which is also directly related to the budget. If we only consider the cost and ignore the reward, it is possible to select fewer users or reduce the user's profit; if we only consider the reward and ignore the cost, the quality of the task cannot be guaranteed. Therefore, the UR-GAT algorithm proposed in this paper comprehensively considers the cost and reward, namely unit reward.

Our algorithm maximizes the total profit while meeting the budget of task. Profit equals the user's reward minus cost, unit reward is equal to the ratio of rewards to costs. The intuition is that the greater the unit reward of users, the greater the user's profit, and vice versa. Compared with Algorithm 2, this algorithm comprehensively considers the costs and rewards of selected users, selects users with higher unit rewards of each type, and calculates the total profit and type diversity of selected users.

On the basis of Algorithm 2, the implementation of the Algorithm 3 is modified. Line 4 is modified to sort users with type j by unit reward in descending order as T_j .

Algorithm 3 Unit Reward-based Greedy Algorithm by Type (UR-GAT).

Require: task τ , user set U , the number of types k .

Ensure: the task assignment results: $\{x_1, x_2, \dots, x_i, \dots, x_m | u_i \in U\}$.

```

1: for all  $u_i \in U$  do
2:   for all type  $j$  do
3:     if  $s_i < d_\tau$  and  $u_i$  belongs to type  $j$  then
4:       Insert  $u_i$  into  $T_j$  by unit reward in descending order;
5:     end if
6:   end for
7: end for
8: while  $j$  do
9:   select the lowest cost user  $u_i$  from  $T_j$ ;
10:  if  $b_\tau \geq r_i$  and  $n_\tau > 0$  and  $t_i \in T_\tau$  then
11:    set  $x_i \leftarrow 1$ ;
12:    set  $b_\tau \leftarrow b_\tau - r_i$ ;
13:    set  $n_\tau \leftarrow n_\tau - 1$ ;
14:    if  $n_\tau \leq 0$  then
15:      break;
16:    end if
17:  end if
18:   $j \leftarrow j + 1$ ;
19:  if  $j == k + 1$  then
20:     $j \leftarrow 1$ ;
21:  end if
22: end while
23:
24: return  $\{x_1, x_2, \dots, x_i, \dots, x_m\}$ .
  
```

3.4 Algorithm performance analysis

According to the task assignment model proposed in this paper, there are one task and m mobile users, in which the task needs to be completed by multiple types of users and one user can only be assigned to one task. The C-GA algorithm selects users based on cost, giving priority to users with lower costs when assigning tasks, without considering the type of user required by the task. Both the C-GAT algorithm and the UR-GAT algorithm give priority to the type of user required by the task, and then the former selects users based on cost, and the latter selects users based on unit reward. When the three heuristic algorithms select users, they are sorted according to cost or unit reward, and the time complexity can be written as $O(m \cdot \log m)$, and the

Table 4 Experimental setup parameters

Parameter	Value
$ U = m$	{20, 30, 40, 50, 60}
c_i	[5, 10]
r_i	[15, 20]
s_i	[15, 24]
d_τ	[21, 24]
$ T_\tau $	{2, 3, 4, 5, 6}
n_τ	[4, 6]
b_τ	{100, 105, 110, 115, 120}
t_i	$U(1, 6)$ or $N(3, (2/3)^2)$ or $E(7/3)$

maximum number of iterations of the three algorithms when selecting users is m , so the total time complexity is $O(m \cdot \log m + m)$, simplified to $O(m \cdot \log m)$.

Because the objective function is to maximize the user type diversity, the users assigned to the task should be distributed as evenly as possible among the user types required by the task. Both the C-GAT algorithm and the UR-GAT algorithm consider the task requirements, so the calculated type The diversity value is better than the C-GA algorithm.

4 Experimental evaluation

In this section, we experimentally verify the user type diversity, total profit and running time of the three proposed algorithms with various number of users, budget of task and the required types of task. The evaluation parameters are shown in Table 4. All computations are performed on a 64-bit PC with Intel(R) Core(TM) i7-7820X 3.6 GHz and 16 GB memory.

The user type t_i follows the Uniform distribution $U(1, |T|)$, or the Normal distribution $N(|T|/2.0, ((|T|/2.0 - 1)/3.0)^2)$, or the Exponential distribution $E((|T| + 1)/3.0)$.

4.1 Experiment on synthetic datasets

In this section, we introduce the user type synthetic datasets. We introduce three distributions: Uniform distribution, Normal distribution and Exponential distribution as the user type distributions.

First, we give the probability density function of each user type distribution.

The Uniform distribution is defined by two parameters a and b , which are the minimum and maximum values on the number axis. Equation 2 shows the probability density function.

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Random variable t_i follows a Normal distribution with mathematical expectation μ and variance σ^2 , denoted as $N(\mu, \sigma^2)$. Equation 3 shows the probability density function.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3)$$

The random variable t_i follows an Exponential distribution with parameter λ , where $\lambda > 0$ is a constant. Equation 4 shows the probability density function.

$$f(x) = \begin{cases} \lambda e^{-\lambda \cdot x}, & x > 0 \\ 0, & otherwise \end{cases} \quad (4)$$

Then, we explain the parameter settings of the three major distributions.

In this experiment, we set left and right boundaries of the Uniform distribute to $a = 1$, $b = |T|$. In the experiment we set $|T| = 6$.

In the Normal distribution, we set $\mu = |T|/2$. According to the 3σ rule in statistics, the probability that a large amount of data falls in that interval is 88.89%. The smallest type in the data set is 1, and the left interval is 1, we can deduce $\sigma = ((|T|/2) - 1)/3$.

In the Exponential distribution, we set $\lambda = (|T| + 1)/3$.

Finally, in our experiment, the user type t_i follows the Uniform distribution $U(1, 6)$, or the Normal distribution $N(3, (2/3)^2)$, or the Exponential distribution $E(7/3)$.

4.2 Evaluation criteria

4.2.1 Three formulas and definitions of type diversity

- *Entropy* In the experimental evaluation we use the entropy to measure the type diversity of the recruited user types as follow equation.

$$- \sum_{j \in T} p_j \cdot \log(p_j) \quad (5)$$

- *Gini coefficient* In the experimental evaluation, we use the Gini coefficient to measure the stability of the heuristic algorithms, the calculation formula is as follows.

$$1 - \sum_{j \in T} p_j^2 \quad (6)$$

- *Error rate* In the experimental evaluation, We use the error rate to measure the stability of the three heuristic algorithms, the calculation formula is as follows.

$$1 - \max \{p_j\} \quad (7)$$

4.2.2 The definition of total profit

- *Total profit* Here, the total profit is the profit sum of all the assigned users after the task assignment. The calculation of the total profit is shown in below formula.

$$\sum_{i=1}^m x_i \cdot (r_i - c_i) \quad (8)$$

Profit is related to cost and reward. The cost of the user performing the task is denoted as c_i , and the reward paid to the user is denoted as r_i . The profit of an individual user refers to the difference between the reward and the cost. If the user u_i is assigned to task τ , then the user's profit can be added to the total revenue, otherwise it cannot be added.

4.2.3 Stability

Variance is a measure of the degree of dispersion in probability theory and statistical variance when measuring random variables or a set of data. Here, we calculate the variance based on the average value of the diversity obtained from the experiment to measure the stability of each algorithm. The following formula 9 shows the calculation of population variance:

$$Var = \frac{\sum (x - \mu)^2}{N} \quad (9)$$

4.2.4 Running time

Running time is often used to measure the operating efficiency of the program. Here, we recorded the execution time of the program and calculated their average, maximum and minimum values. In this way, you can intuitively see the operating efficiency of the three algorithms under different data changes.

4.3 Analysis on diversity and stability

4.3.1 The impact of budget changes on diversity and stability

Table 5 shows that the type diversity of all algorithms with a varying budget of task, when the required type of task and the number of users are fixed as 4 and 60 respectively. the user type t_i follows the Uniform distribution $U(1, 6)$, or the Normal distribution $N(3, (2/3)^2)$, or the Exponential distribution $E(7/3)$. For simplicity, in Table 5, **UD**, **ND**, and **ED** denote Uniform distribution, Normal distribution and Exponential distribution.

As seen, the C-GAT algorithm and UR-GAT algorithm achieve better type diversity with the increase of the budget of task, whereas the C-GA is insensitive to the budget of task. This observation can be easily interpreted as follows. For the C-GAT algorithm

Table 5 The impact of budget changes on user type diversity ($|T| = 6, m = 60, |T_\tau| = 4$)

		Entropy			Gini			Error			Var	
		UD	ND	ED	UD	ND	ED	UD	ND	ED		
$b_\tau = 100$	C-GA	1.0310	0.8036	0.6582	0.6210	0.4910	0.4270	0.5400	0.3700	0.3300	0.0501	0.0501
	C-GAT	1.3430	1.0380	1.2182	0.7260	0.6280	0.6850	0.6300	0.5800	0.5800	0.0868	0.0868
	UR-GAT	1.3430	1.0380	1.2182	0.7260	0.6280	0.6850	0.6300	0.5800	0.5800	0.0868	0.0868
$b_\tau = 105$	C-GA	0.9485	0.8308	0.6328	0.5830	0.5490	0.4420	0.5100	0.5000	0.3600	0.0349	0.0349
	C-GAT	1.3646	0.9765	1.1567	0.7380	0.6060	0.6630	0.6900	0.5400	0.5900	0.0824	0.0824
	UR-GAT	1.3646	0.9765	1.1567	0.7380	0.6060	0.6630	0.6900	0.5400	0.5900	0.0824	0.0824
$b_\tau = 110$	C-GA	1.0100	0.6980	1.0793	0.6050	0.4680	0.6300	0.5000	0.3800	0.5500	0.0571	0.0571
	C-GAT	1.3430	0.9826	1.3430	0.7260	0.6120	0.7260	0.6300	0.5800	0.6300	0.0950	0.0950
	UR-GAT	1.3430	0.9826	1.3430	0.7260	0.6120	0.7260	0.6300	0.5800	0.6300	0.0950	0.0950
$b_\tau = 115$	C-GA	1.0464	0.7673	0.6582	0.6030	0.4930	0.4270	0.4900	0.3800	0.3300	0.0501	0.0501
	C-GAT	1.3430	0.8902	1.2876	0.7260	0.5970	0.7100	0.6300	0.5000	0.6300	0.0927	0.0927
	UR-GAT	1.3430	0.8902	1.2876	0.7260	0.5970	0.7100	0.6300	0.5000	0.6300	0.0927	0.0927
$b_\tau = 120$	C-GA	1.1240	0.6939	0.7013	0.6400	0.4640	0.4320	0.5200	0.3600	0.3200	0.0603	0.0603
	C-GAT	1.3322	0.8932	1.2767	0.7200	0.6000	0.7040	0.6000	0.5200	0.6000	0.0916	0.0916
	UR-GAT	1.3322	0.8932	1.2767	0.7200	0.6000	0.7040	0.6000	0.5200	0.6000	0.0916	0.0916

and UR-GAT algorithm, they select users in turn for each required type of task, so they have the same trend. This is because C-GA only considers the cost of the user to perform the task and does not consider the required type for the task, and the size of the type diversity is related to the type distribution of the users selected by the task. However, the C-GA achieves the worst performance in terms of type diversity, and the result is relatively unstable with the increase of the budget of task, this is because the algorithm randomly selects the lowest cost user.

From Table 5, we can find that the both proposed UR-GAT algorithm and C-GAT algorithm have achieved the same user type diversity and they are unaffected by different user type distributions when budget changes.

We used information entropy, Gini coefficient, and error rate as the objective functions, and completed 45 comparisons by changing the size of the budget. Experimental simulation results show that in terms of type diversity, when the types of users belong to different distributions and change the objective function, as the budget increases, the performance of the three algorithms has not changed, namely C-GAT algorithm and UR-GAT algorithm is superior to C-GA algorithm. In terms of data sensitivity, the C-GA algorithm is superior to the other two algorithms.

4.3.2 The impact of required type changes on diversity and stability

When the budget of the task and the number of users are fixed, the type diversity increases substantially with the increase of the required type of task, as shown in Table 6, when the budget of task and the number of users are fixed as 115 and 60 respectively. Here, the user type t_i follows the Uniform distribution $U(1, 6)$, or the Normal distribution $N(3, (2/3)^2)$, or the Exponential distribution $E(7/3)$. As expected, we can see that the C-GAT algorithm and UR-GAT algorithm achieve the highest performance in terms of the type diversity, and observe that the result is relatively stable with the increase of required type of task. This is because the value of type diversity is closely related to the required type by task. The more types required, the greater the type diversity of UR-GAT and C-GAT.

From Table 6, we can find that the both proposed UR-GAT algorithm and C-GAT algorithm have achieved the same user type diversity and they are unaffected by different user type distributions when the number of required type changes.

We used information entropy, Gini coefficient, and misclassification rate as the objective functions, respectively, and completed 45 comparisons by changing the number of types required by the task. Experimental simulation results show that in terms of type diversity, when the types of users belong to different distributions and change the objective function, as the number of types required by the task increases, the performance of the three algorithms does not change, that is, C-GAT algorithm and UR-GAT algorithm are superior to C-GA algorithm. In terms of data sensitivity, the C-GA algorithm outperforms the other two algorithms in most cases.

4.3.3 The impact of the number of users changes on diversity and stability

In Table 7, we change the number of users to test the type diversity and stability of all algorithms where the user type t_i follows the Uniform distribution $U(1, 6)$, or the

Table 6 The impact of required type changes on user type diversity ($|T| = 6$, $b_\tau = 115$, $m = 60$)

		Entropy			Gini			Error			Var	
		UD	ND	ED	UD	ND	ED	UD	ND	ED		
$ T_\tau = 2$	C-GA	0.6164	0.4266	0.5735	0.4270	0.5660	0.3880	0.3300	0.3800	0.3000	0.0129	
	C-GAT	0.6770	0.4611	0.6770	0.4840	0.5980	0.4840	0.4200	0.4200	0.4200	0.0114	
	UR-GAT	0.6770	0.4611	0.6770	0.4840	0.5980	0.4840	0.4200	0.4200	0.4200	0.0114	
$ T_\tau = 3$	C-GA	0.7567	0.6850	0.8467	0.4800	0.4880	0.5280	0.4000	0.3600	0.4400	0.0286	
	C-GAT	1.0549	0.7405	1.0549	0.6400	0.5360	0.6400	0.6000	0.4400	0.6000	0.0469	
	UR-GAT	1.0549	0.7405	1.0549	0.6400	0.5360	0.6400	0.6000	0.4400	0.6000	0.0469	
$ T_\tau = 4$	C-GA	0.9546	0.7346	1.2182	0.5570	0.4590	0.6850	0.4200	0.3700	0.5800	0.0754	
	C-GAT	1.3430	1.1073	1.3430	0.7260	0.6530	0.7260	0.6300	0.5800	0.6300	0.0988	
	UR-GAT	1.3430	1.1073	1.3430	0.7260	0.6530	0.7260	0.6300	0.5800	0.6300	0.0988	
$ T_\tau = 5$	C-GA	1.1018	0.8382	0.7900	0.6190	0.5230	0.4750	0.4900	0.4100	0.3700	0.0578	
	C-GAT	1.5648	1.0519	1.5648	0.7900	0.6370	0.7900	0.7900	0.5800	0.7900	0.1380	
	UR-GAT	1.5648	1.0519	1.5648	0.7900	0.6370	0.7900	0.7900	0.5800	0.7900	0.1380	
$ T_\tau = 6$	C-GA	1.1794	0.8048	1.1449	0.6560	0.5280	0.6240	0.5600	0.4400	0.5600	0.0723	
	C-GAT	1.6094	1.3876	1.6094	0.8000	0.7360	0.8000	0.8000	0.6800	0.8000	0.1525	
	UR-GAT	1.6094	1.3876	1.6094	0.8000	0.7360	0.8000	0.8000	0.6800	0.8000	0.1525	

Table 7 The impact of the number of users changes on user type diversity ($|T| = 6, b_t = 115, |T_-| = 4$)

		Entropy			Gini			Error			Var
		UD	ND	ED	UD	ND	ED	UD	ND	ED	
m = 20	C-GA	0.9324	0.7713	0.9306	0.5680	0.4970	0.5700	0.4700	0.4000	0.4400	0.0426
	C-GAT	1.3538	0.9031	1.0972	0.7320	0.5770	0.6430	0.6600	0.5200	0.5700	0.0792
	UR-GAT	1.3538	0.9031	1.0972	0.7320	0.5770	0.6430	0.6600	0.5200	0.5700	0.0792
m = 30	C-GA	1.0894	0.6113	0.9576	0.6400	0.3840	0.5920	0.5600	0.2800	0.5200	0.0650
	C-GAT	1.3322	0.8932	1.2767	0.7200	0.6000	0.7040	0.6000	0.5200	0.6000	0.0916
	UR-GAT	1.3322	0.8932	1.2767	0.7200	0.6000	0.7040	0.6000	0.5200	0.6000	0.0916
m = 40	C-GA	0.9140	0.5982	0.7039	0.5510	0.4100	0.4330	0.4700	0.3200	0.3100	0.0378
	C-GAT	1.2328	0.9001	1.3646	0.6900	0.5740	0.7380	0.6500	0.5500	0.6900	0.0845
	UR-GAT	1.2328	0.9001	1.3646	0.6900	0.5740	0.7380	0.6500	0.5500	0.6900	0.0845
m = 50	C-GA	0.8601	0.6676	0.9485	0.5400	0.4350	0.5830	0.4500	0.3200	0.5100	0.0416
	C-GAT	1.3646	1.1706	1.3646	0.7380	0.6720	0.7380	0.6900	0.5900	0.6900	0.0991
	UR-GAT	1.3646	1.1706	1.3646	0.7380	0.6720	0.7380	0.6900	0.5900	0.6900	0.0991
m = 60	C-GA	0.9576	0.6385	0.7494	0.5920	0.4480	0.4800	0.5200	0.3600	0.3600	0.0377
	C-GAT	1.3322	0.9576	1.2213	0.7200	0.5920	0.6880	0.6000	0.5200	0.6000	0.0883
	UR-GAT	1.3322	0.9576	1.2213	0.7200	0.5920	0.6880	0.6000	0.5200	0.6000	0.0883

Normal distribution $N(3, (2/3)^2)$, or the Exponential distribution $E(7/3)$. Here, the budget of task and the number of required types are fixed as 115 and 4 respectively. Clearly, C-GAT algorithm and UR-GAT algorithm are far superior to the C-GA algorithm. It can be seen from the table that the change trend of type diversity value in C-GAT algorithm and UR-GAT algorithm is the same, because they select users according to the type of task required, and the selected users are evenly distributed among the type required for the task. Conversely, the type diversity value of C-GA is always lower than the other two algorithms, because the algorithm selects users at random.

From Table 7, we can find that the both proposed UR-GAT algorithm and C-GAT algorithm have achieved the same user type diversity and they are unaffected by different user type distributions when the number of users changes.

We used entropy, gini coefficient, and error rate as the objective functions, and completed 45 comparisons by changing the number of users. Experimental simulation results show that in terms of type diversity, the C-GAT algorithm and the UR-GAT algorithm are superior to the C-GA algorithm. In terms of data sensitivity, the C-GA algorithm is superior to the other two algorithms.

4.4 Analysis on total profit

4.4.1 The impact of budget changes on total profit

Total profit refers to the sum of the profit of all selected users. As the task budget increases, the total profit of the three algorithms changes as shown Table 8. Here, the required type of task and the number of users are fixed as 4 and 60 respectively. The user type t_i follows three different distributions, namely Uniform distribution $U(1, 6)$, Normal distribution $N(3, (2/3)^2)$, and Exponential distribution $E(7/3)$. It can be observed that the total profit of UR-GAT algorithm is significantly better than the other two algorithms, because the algorithm considers the unit reward. The larger the unit reward, the greater the value of total profit.

After 15 comparisons in this group of experiments, it can be clearly observed that the simulation results are accidental. In terms of total profit, the G-GA algorithm is sometimes better, but on average, the UR-GAT algorithm is better. As the number of types required by the task increases, the stability of the G-GA algorithm is superior to the other two algorithms.

4.4.2 The impact of required type changes on total profit

With the increase of task required types, the overall trend of the three algorithms is the same, as shown in Table 9. The user type t_i follows three different distributions, namely Uniform distribution $U(1, 6)$, Normal distribution $N(3, (2/3)^2)$, and Exponential distribution $E(7/3)$. The budget of task and the number of users are fixed as 115 and 60 respectively. With the increase of the required type of task, although the total profit of UR-GAT achieves the best performance, its changing trend is also unstable. It is because that total profit is unrelated to the types of users.

Table 8 The impact of budget changes on total profit ($|T| = 6, m = 60, |T_\tau| = 4$)

Entropy										
		UD			ND			ED		
		Average	Min	Max	Average	Min	Max	Average	Min	Max
$b_\tau = 100$	C-GA	58.2	49.0	65.0	58.8	49.0	65.0	56.6	50.0	60.0
	C-GAT	56.8	50.0	61.0	59.0	50.0	65.0	54.6	50.0	58.0
	UR-GAT	58.8	54.0	61.0	62.2	52.0	67.0	58.0	55.0	63.0
$b_\tau = 105$	C-GA	50.0	44.0	61.0	51.0	44.0	61.0	50.8	44.0	59.0
	C-GAT	48.8	43.0	58.0	50.2	45.0	61.0	45.0	41.0	49.0
	UR-GAT	52.0	45.0	60.0	53.0	47.0	62.0	50.4	46.0	56.0
$b_\tau = 110$	C-GA	57.8	44.0	63.0	59.4	47.0	65.0	58.2	44.0	63.0
	C-GAT	57.2	45.0	63.0	57.0	47.0	63.0	56.4	44.0	63.0
	UR-GAT	62.2	50.0	68.0	61.6	53.0	67.0	61.4	53.0	68.0
$b_\tau = 115$	C-GA	58.0	44.0	63.0	54.4	47.0	63.0	57.4	44.0	63.0
	C-GAT	57.8	47.0	64.0	51.4	39.0	62.0	56.0	47.0	64.0
	UR-GAT	61.0	50.0	67.0	52.0	39.0	62.0	60.0	47.0	65.0
$b_\tau = 120$	C-GA	61.8	60.0	63.0	57.4	47.0	65.0	61.0	59.0	63.0
	C-GAT	60.8	54.0	64.0	54.8	39.0	62.0	57.0	52.0	64.0
	UR-GAT	64.6	59.0	68.0	55.4	39.0	65.0	61.6	59.0	65.0

Table 9 The impact of required type changes on total profit ($|T| = 6$, $b_T = 115$, $m = 60$)

		Entropy			ND			ED		
		UD								
		Average	Min	Max	Average	Min	Max	Average	Min	Max
$ T_r = 2$	C-GA	56.8	45.0	65.0	39.8	24.0	62.0	56.0	39.0	62.0
	C-GAT	56.8	45.0	66.0	39.6	24.0	61.0	56.0	39.0	64.0
	UR-GAT	58.0	45.0	67.0	39.0	24.0	62.0	57.6	39.0	67.0
$ T_r = 3$	C-GA	60.4	57.0	64.0	57.4	42.0	64.0	60.2	58.0	63.0
	C-GAT	59.6	57.0	64.0	57.8	42.0	63.0	60.4	60.0	61.0
	UR-GAT	61.8	57.0	68.0	59.2	42.0	66.0	63.6	60.0	68.0
$ T_r = 4$	C-GA	57.6	55.0	60.0	57.0	51.0	62.0	55.2	46.0	59.0
	C-GAT	56.2	47.0	62.0	55.4	49.0	61.0	54.4	40.0	59.0
	UR-GAT	59.2	54.0	62.0	58.6	52.0	67.0	55.4	40.0	64.0
$ T_r = 5$	C-GA	59.0	52.0	65.0	59.4	53.0	65.0	58.4	50.0	65.0
	C-GAT	58.2	46.0	66.0	57.8	54.0	64.0	54.6	45.0	63.0
	UR-GAT	61.4	50.0	69.0	59.8	54.0	65.0	58.8	48.0	68.0
$ T_r = 6$	C-GA	61.2	52.0	67.0	61.2	52.0	67.0	61.2	52.0	67.0
	C-GAT	60.6	57.0	65.0	56.6	52.0	61.0	59.4	52.0	67.0
	UR-GAT	63.8	60.0	68.0	61.6	58.0	63.0	63.4	58.0	67.0

Table 10 The impact of the number of users changes on total profit ($|T| = 6$, $b_\tau = 115$, $|T_\tau| = 4$)

		Entropy								
		UD			ND			ED		
		Average	Min	Max	Average	Min	Max	Average	Min	Max
m = 20	C-GA	51.8	45.0	58.0	54.4	44.0	64.0	54.2	45.0	66.0
	C-GAT	47.8	42.0	55.0	53.2	44.0	61.0	52.0	39.0	65.0
	UR-GAT	51.6	46.0	58.0	54.8	44.0	63.0	53.8	40.0	66.0
m = 30	C-GA	59.8	59.0	62.0	57.8	53.0	62.0	57.2	54.0	60.0
	C-GAT	59.0	57.0	61.0	55.0	43.0	60.0	55.0	52.0	57.0
	UR-GAT	60.2	57.0	62.0	57.6	45.0	62.0	56.8	52.0	60.0
m = 40	C-GA	55.4	46.0	63.0	53.4	44.0	63.0	53.4	43.0	63.0
	C-GAT	54.8	47.0	62.0	50.4	44.0	59.0	50.4	41.0	63.0
	UR-GAT	55.8	47.0	65.0	52.4	44.0	65.0	53.0	46.0	66.0
m = 50	C-GA	52.6	41.0	65.0	54.4	49.0	65.0	50.4	43.0	58.0
	C-GAT	50.8	45.0	63.0	50.8	46.0	58.0	48.2	41.0	55.0
	UR-GAT	55.2	48.0	63.0	51.6	47.0	58.0	51.6	45.0	59.0
m = 60	C-GA	60.8	57.0	64.0	59.8	54.0	63.0	61.6	58.0	65.0
	C-GAT	59.4	54.0	63.0	57.0	54.0	61.0	63.4	62.0	64.0
	UR-GAT	63.6	59.0	68.0	62.0	55.0	68.0	64.2	63.0	65.0

After 15 comparisons in this group of experiments, only one comparison showed that the C-GA algorithm achieved better performance. Therefore, in terms of total profit, it can be clearly observed that the simulation results are accidental. The C-GA algorithm is sometimes better, but the average of the total profit is better from the UR-GAT algorithm.

4.4.3 The impact of the number of users changes on total profit

In Table 10, we change the number of users to evaluate the total profit of all algorithms. The user type t_i follows three different distributions, namely Uniform distribution $U(1, 6)$, Normal distribution $N(3, (2/3)^2)$, and Exponential distribution $E(7/3)$. Here, the budget of task and the number of required types are fixed as 115 and 4 respectively. Obviously, UR-GAT algorithm is far superior to the C-GA algorithm and C-GAT algorithm. According to the Table 10, the total profit of UR-GAT is always higher than the other two algorithms, because this algorithm comprehensively considers the costs and rewards.

After 15 comparisons in this group of experiments, it can be clearly observed that the simulation results are accidental in terms of total profit. The C-GA algorithm is sometimes better, but the UR-GAT algorithm is better from the average of total profit. It can be seen that the total profit of C-GA is sometimes higher than that of UR-GAT. This is because the algorithm only considers the cost and does not consider the type of users. Because there are fewer restrictions, the range of user choices is expanded.

4.5 Analysis on running time

4.5.1 The impact of budget changes on running time

Running time refers to the execution time of the program. Table 11 shows the running time of all algorithms with various budget of task when the required type of task and the number of users are fixed as 4 and 60 respectively. As can be seen from the above table, when the user type t_i follows the Uniform distribution $U(1, 6)$, or the Normal distribution $N(3, (2/3)^2)$, or the Exponential distribution $E(7/3)$, compared with C-GA algorithm the running time of C-GAT algorithm and UR-GAT algorithm is much longer. It can be seen that the operating efficiency of C-GA algorithm is significantly better than the other two algorithms.

4.5.2 The impact of required type changes on running time

In Table 12, we change the required type of task to test the running time of all algorithms. Here, the task budget and the number of users are fixed as 115 and 60 respectively. As expected, we can see that C-GA algorithm achieves the highest performance in terms of operating efficiency, and observes that the results are relatively stable with the increase in the required type of task, without significant changes. The running time of C-GAT algorithm and UR-GAT algorithm is not much different, and the efficiency is about the same. It can be seen that the operating efficiency of C-GA algorithm is significantly better than the other two algorithms.

4.5.3 The impact of the number of users changes on running time

In Table 13, we change the user number to test the running time of all algorithms. The user type t_i follows the Uniform distribution $U(1, 6)$, or the Normal distribution $N(3, (2/3)^2)$, or the Exponential distribution $E(7/3)$. Here, the task budget and the number of required types are fixed at 115 and 4 respectively. Obviously, we conclude that C-GA algorithm runs far better than C-GAT algorithm and UR-GAT algorithm.

From the above analysis, it can be concluded that no matter what kind of variable changes, the operating efficiency of C-GA algorithm is better than C-GAT algorithm and UR-GAT algorithm, and C-GA algorithm is more stable, and the float is not too large. The C-GAT algorithm and UR-GAT algorithm will fluctuate up and down.

5 Conclusion

In this paper, we propose a novel type diversity-oriented task assignment problem under the constraints, such as the budget, the latest ending time, the required types and the required number of users. Then, we formalize the task assignment problem. Moreover, we propose three heuristic algorithms to maximize the type diversity with budget constraint. If the users selected to perform the task are distributed with equal probability in each type required for the task, then the goal of task assignment optimization is to maximize the user type diversity. When different objective functions are used to

Table 11 The impact of budget changes on running time ($|T| = 6$, $m = 60$, $|T_-| = 4$)

	Entropy							
	UD			ND			ED	
	Ave	Min	Max	Ave	Min	Max	Ave	Max
$b_\tau = 100$	C-GA	0.0404	0.0366	0.0508	0.0373	0.0363	0.0374	0.0382
	C-GAT	0.1850	0.1286	0.2933	0.0980	0.0464	0.1496	0.2052
	UR-GAT	0.1872	0.1309	0.2635	0.1055	0.0472	0.1494	0.1846
$b_\tau = 105$	C-GA	0.0397	0.0358	0.0506	0.0374	0.0364	0.0368	0.0372
	C-GAT	0.1801	0.1453	0.2346	0.0819	0.0384	0.1053	0.1313
	UR-GAT	0.1809	0.1490	0.2184	0.0820	0.0406	0.1111	0.1445
$b_\tau = 110$	C-GA	0.0411	0.0366	0.0567	0.0385	0.0364	0.0395	0.0429
	C-GAT	0.1947	0.1602	0.2521	0.0780	0.0469	0.1151	0.1356
	UR-GAT	0.1947	0.1354	0.2589	0.0770	0.0464	0.1182	0.1454
$b_\tau = 115$	C-GA	0.0414	0.0376	0.0514	0.0433	0.0386	0.0391	0.0404
	C-GAT	0.1356	0.0868	0.1658	0.0968	0.0626	0.1216	0.1752
	UR-GAT	0.1457	0.1104	0.1653	0.1009	0.0646	0.1201	0.1542
$b_\tau = 120$	C-GA	0.0487	0.0404	0.0608	0.0418	0.0402	0.0426	0.0444
	C-GAT	0.1430	0.1143	0.1643	0.0849	0.0644	0.1069	0.1927
	UR-GAT	0.1333	0.0941	0.1661	0.0850	0.0660	0.1156	0.1785

Table 12 The impact of required type changes on running time ($|T| = 6, b_T = 115, m = 60$)

	Entropy								
	UD			ND			ED		
	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max
$ T_r = 2$	C-GA	0.0438	0.0379	0.0585	0.0405	0.0379	0.0447	0.0379	0.0523
	C-GAT	0.1188	0.1011	0.1675	0.0698	0.0565	0.0778	0.0915	0.1436
	UR-GAT	0.1095	0.0805	0.1625	0.0681	0.0423	0.0935	0.0722	0.1491
$ T_r = 3$	C-GA	0.0535	0.0400	0.0689	0.0416	0.0377	0.0490	0.0378	0.0433
	C-GAT	0.1328	0.0790	0.1859	0.0860	0.0622	0.1060	0.0737	0.1435
	UR-GAT	0.1010	0.0386	0.1369	0.0896	0.0581	0.1231	0.0892	0.1515
$ T_r = 4$	C-GA	0.0429	0.0384	0.0526	0.0405	0.0386	0.0438	0.0397	0.0499
	C-GAT	0.1376	0.0447	0.1828	0.1121	0.0812	0.1709	0.0895	0.2626
	UR-GAT	0.1344	0.0399	0.2155	0.1193	0.0894	0.1681	0.1073	0.1927
$ T_r = 5$	C-GA	0.0451	0.0381	0.0595	0.0393	0.0381	0.0420	0.0375	0.0438
	C-GAT	0.1257	0.1061	0.1551	0.0749	0.0539	0.1089	0.0938	0.2109
	UR-GAT	0.1268	0.0957	0.1501	0.0771	0.0552	0.1195	0.0927	0.1572
$ T_r = 6$	C-GA	0.0401	0.0376	0.0459	0.0390	0.0375	0.0408	0.0370	0.0421
	C-GAT	0.1766	0.1043	0.2538	0.1255	0.0703	0.1828	0.0818	0.2646
	UR-GAT	0.1763	0.1122	0.2684	0.1490	0.0759	0.2004	0.0843	0.2809

Table 13 The impact of the number of users changes on running time ($|T| = 6$, $b_T = 115$, $|T_T| = 4$)

		Entropy								
		UD			ND			ED		
		Ave	Min	Max	Ave	Min	Max	Ave	Min	Max
m = 20	C-GA	0.0403	0.0324	0.0588	0.0349	0.0333	0.0384	0.0400	0.0338	0.0527
	C-GAT	0.0949	0.0511	0.1301	0.0727	0.0524	0.0832	0.0921	0.0587	0.1233
	UR-GAT	0.1054	0.0518	0.1546	0.0770	0.0560	0.0919	0.0915	0.0568	0.1246
m = 30	C-GA	0.0434	0.0378	0.0529	0.0381	0.0372	0.0398	0.0375	0.0343	0.0396
	C-GAT	0.1194	0.0605	0.1848	0.0757	0.0586	0.1152	0.1107	0.0637	0.1596
	UR-GAT	0.1141	0.0622	0.1730	0.0726	0.0593	0.0898	0.1089	0.0636	0.1573
m = 40	C-GA	0.0425	0.0374	0.0521	0.0417	0.0393	0.0500	0.0384	0.0374	0.0410
	C-GAT	0.1228	0.0634	0.1684	0.0720	0.0647	0.0839	0.1228	0.0738	0.1538
	UR-GAT	0.1207	0.0639	0.1640	0.0844	0.0713	0.1056	0.1233	0.0755	0.1632
m = 50	C-GA	0.0499	0.0408	0.0667	0.0422	0.0372	0.0501	0.0400	0.0373	0.0431
	C-GAT	0.1352	0.0717	0.2572	0.1039	0.0655	0.1269	0.0937	0.0394	0.1444
	UR-GAT	0.1307	0.0750	0.2142	0.0988	0.0693	0.1370	0.1361	0.0816	0.2635
m = 60	C-GA	0.0444	0.0380	0.0633	0.0405	0.0382	0.0435	0.0410	0.0381	0.0430
	C-GAT	0.1235	0.0712	0.1872	0.0881	0.0666	0.1248	0.1101	0.0705	0.1471
	UR-GAT	0.1317	0.0753	0.2063	0.0902	0.0692	0.1271	0.1179	0.0756	0.1569

evaluate the stability of the three algorithms and the user types are subject to different distribution functions, as far as type diversity and total profit are concerned, further simulation experiments show that the performance of the algorithm is not affected, that is, UR-GAT algorithm is superior to C-GA algorithm and C-GAT algorithm; in terms of running time, simulation experiment results show that C-GA algorithm is superior to the other two algorithms.

Acknowledgements This work is partly supported by the National Natural Science Foundation of China under Grant Nos. 61977044, 61872228 and 61877037, the National Key R&D Program of China under Grant No. 2017YFB1402102, the Key R&D Program of Shaanxi Province under Grant No. 2020GY-221, 2019ZDLSF07-01, 2020ZDLGY10-05, the Natural Science Basis Research Plan in Shaanxi Province of China under Grant Nos. 2020JM-303, 2020JM-302, 2017JM6060, the Fundamental Research Funds for the Central Universities of China under Grant No. GK201903090, GK201801004, the S&T Plan of Xi'an City of China under Grant No. 2019216914GXRC005CG006-GXYD5.1, the Shaanxi Normal University Foundational Education Course Research Center of Ministry of Education of China under Grant No. 2019-JCJY009.

References

- Abououf M, Mizouni R, Singh S, Otok H, Ouali A (2019) Multi-worker multi-task selection framework in mobile crowd sourcing. *J Netw Comput Appl* 130:52–62
- Alhamid M, Rawashdeh M, Dong H, Hossain M, Saddik A (2016) Exploring latent preferences for context-aware personalized recommendation systems. *IEEE Trans Hum Mach Syst* 46(4):615–623
- Cai Z, Duan Z, Li W (2020) Exploiting multi-dimensional task diversity in distributed auctions for mobile crowdsensing. *IEEE Trans Mob Comput*. <https://doi.org/10.1109/TMC.2020.2987881>
- Cheng P, Lian X, Chen Z, Chen L, Han J, Zhao J (2015a) Reliable diversity-based spatial crowdsourcing by moving workers. *Proc VLDB Endow* 8(10):1022–1033
- Cheng P, Lian X, Chen L, Han J, Zhao J (2015b) Task assignment on multi-skill oriented spatial crowdsourcing. *IEEE Trans Knowl Data Eng* 28(8):2201–2215
- Cohen S, Yashinski M (2017) Crowdsourcing with diverse groups of users. In: *Proceeding WebDB'17 proceedings of the 20th international workshop on the web and databases*. ACM, Chicago, IL, USA, pp 7–12
- Cranshaw J, Toch E, Hong J, Kittur A, Sadeh N (2010) Bridging the gap between physical location and online social networks. In: *Proceeding of the 12th ACM international conference on ubiquitous computing*. ACM, Copenhagen, Denmark, pp 119–128
- Duan Z, Li W, Cai Z (2017) Distributed auctions for task assignment and scheduling in mobile crowdsensing systems. In: *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE Computer Society, Atlanta, GA, USA, pp 635–644
- Duan Z, Li W, Zheng X, Cai Z (2019) Mutual-preference driven truthful auction mechanism in mobile crowdsensing. In: *The 39th IEEE international conference on distributed computing systems (ICDCS)*. IEEE, Dallas, Texas, USA
- Ganti RK, Ye F, Lei H (2011) Mobile crowdsensing: current state and future challenges. *IEEE Commun Mag* 49(11):32–49
- Gong W, Zhang B, Li C (2019) Location-based online task assignment and path planning for mobile crowdsensing. *IEEE Trans Veh Technol* 68(2):1772–1783
- Kazemi L, Shahabi C (2012) GeoCrowd: enabling query answering with spatial crowdsourcing. In: *International conference on advances in geographic information systems*. ACM, Redondo Beach, CA, USA, pp 189–198
- Kong C, Luo G, Tian L, Cao X (2019) Disseminating authorized content via data analysis in opportunistic social networks. *Big Data Min Anal* 2(1):12–24
- Li J, Cai Z, Yan M, Li Y (2016) Using Crowdsourced data in location-based social networks to explore influence maximization. In: *The 35th annual IEEE international conference on computer communications (INFOCOM)*. IEEE, San Francisco, CA, USA, pp 1–9

- Li J, Cai Z, Wang J, Han M, Li Y (2018a) Truthful incentive mechanisms for geographical position conflicting mobile crowdsensing systems. *IEEE Trans Comput Soc Syst* 5(2):324–334
- Li L, Zhang L, Wang X, Yu S, Wang A (2018b) An efficient task allocation scheme with capability diversity in crowdsensing. In: China conference on wireless sensor networks (CWSN). Springer, Kunming, China, pp 12–20
- Li M, Zheng Y, Jin X, Guo C (2018c) Task assignment for simple tasks with small budget in mobile crowdsourcing. In: 2018 14th international conference on mobile ad-hoc and sensor networks (MSN). IEEE, Shenyang, China, pp 68–73
- Li Y, Jiang Y, Wu W, Jiang J, Fan H (2019) Room allocation with capacity diversity and budget constraints. *IEEE Access* 7:42968–42986
- Liao J, Tang J, Zhao X (2019) Course drop-out prediction on MOOC platform via clustering and tensor completion. *Tsinghua Sci Technol* 24(4):44–54
- Liu T, Wang Y, Li Y, Tong X, Qi L, Jiang N (2020) Privacy protection based on stream cipher for spatio-temporal data in IoT. *IEEE Internet Things J*. <https://doi.org/10.1109/JIOT.2020.2990428>
- Qiao L, Tang F, Liu J (2018) Feedback based high-quality task assignment in collaborative crowdsourcing. In: 2018 IEEE 32nd international conference on advanced information networking and applications (AINA). IEEE, Krakow, Poland, pp 1139–1146
- Tao X, Song W (2019) Location-dependent task allocation for mobile crowdsensing with clustering effect. *IEEE Internet Things J* 6(1):1029–1045
- Tong Y, Zeng Y, Ding B, Wang L, Chen L (2019) Two-sided online micro-task assignment in spatial crowdsourcing. *IEEE Trans Knowl Data Eng*. <https://doi.org/10.1109/TKDE.2019.2948863>
- Wang X, Wang S (2016) An optimal assignment for mobile sensing tasks in spatial crowdsourcing. In: 2016 5th international conference on computer science and network technology (ICCSNT). IEEE, Changchun, China, pp 681–687
- Wang Y, Cai Z, Ying G, Gao Y, Tong X, Wu G (2016) An incentive mechanism with privacy protection in mobile crowdsourcing systems. *Comput Netw* 102:157–171
- Wang Y, Cai Z, Tong X, Gao Y, Yin G (2018a) Truthful incentive mechanism with location privacy-preserving for mobile crowdsourcing systems. *Comput Netw* 135:32–43
- Wang X, Jia R, Tian X, Gan X (2018b) Dynamic task assignment in crowdsensing with location awareness and location diversity. In: IEEE INFOCOM 2018—IEEE conference on computer communications. IEEE, Honolulu, HI, USA, pp 2420–2428
- Wang L, Yu Z, Han Q, Guo B, Xiong H (2018c) Multi-objective optimization based allocation of heterogeneous spatial crowdsourcing tasks. *IEEE Trans Mob Comput* 17(7):1637–1650
- Wang Y, Cai Z, Zhan Z, Gong Y, Tong X (2019) An optimization and auction based incentive mechanism to maximize social welfare for mobile crowdsourcing. *IEEE Trans Comput Soc Syst* 6(3):414–429
- Wang Y, Gao Y, Li Y, Tong X (2020) A worker-selection incentive mechanism for optimizing platform-centric mobile crowdsourcing systems. *Comput Netw*. <https://doi.org/10.1016/j.comnet.2020.107144>
- Wei G, Zhang B, Cheng L (2018) Task assignment in mobile crowdsensing: present and future directions. *IEEE Netw* 32(4):100–107
- Xing Y, Wang L, Li Z, Zhan Y (2019) Multi-attribute crowdsourcing task assignment with stability and satisfactory. *IEEE Access* 7:133351–133361
- Yao C (2017) Constructing a user-friendly and smart ubiquitous personalized learning environment by using a context-aware mechanism. *IEEE Trans Learn Technol* 10(1):104–114
- Yin X, Chen Y, Li B (2017) Task assignment with guaranteed quality for crowdsourcing platforms. In: 2017 IEEE/ACM 25th international symposium on quality of service. IEEE, Vilanova i la Geltru, Spain, pp 1–10
- Yuan H, Cao P (2019) Collaborative assessments in computer science education: a survey. *Tsinghua Sci Technol* 24(4):435–445
- Yu J, Xiao M, Gao G, Hu H (2016) Minimum cost spatial–temporal task allocation in mobile crowdsensing. In: International conference on wireless algorithms, systems, and applications. Springer, Bozeman, MT, USA, pp 262–271
- Zhang M, Yang P, Tian C, Tang S, Gao X, Wang B, Xiao F (2016) Quality-aware sensing coverage in budget-constrained mobile crowdsensing networks. *IEEE Trans Veh Technol* 65(9):7698–7707
- Zhang Y, Zhang D, Li Q, Wang D (2018) Towards optimized online task allocation in cost-sensitive crowdsensing applications. In: 2018 IEEE 37th international performance computing and communications conference (IPCCC). IEEE, Orlando, FL, USA, pp 1–8

Zhu S, Cai Z, Hu H, Li Y, Li W (2019) ZkCrowd: a hybrid blockchain-based crowdsourcing platform. *IEEE Trans Ind Inform (TII)* 16(6):4196–4205

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.