sensors

*Article*

# Using Greedy Random Adaptive Procedure to Solve the User Selection Problem in Mobile Crowdsourcing

**Jian Yang [1], Xiaojuan Ban [1,*] and Chunxiao Xing [2]**

[1] School of Computer and Communication Engineering, Beijing Key Laboratory of Knowledge Engineering for Materials Science, University of Science and Technology Beijing, Beijing 100083, China

[2] Research Institute of Information, Beijing National Research Center for Information Science and Technology, Department of Computer Science and Technology, Institute of Internet Industry, Tsinghua University, Beijing 100084, China

* Correspondence: banxj@ustb.edu.cn

Received: 24 June 2019; Accepted: 16 July 2019; Published: 18 July 2019

**Abstract:** With the rapid development of mobile networks and smart terminals, mobile crowdsourcing has aroused the interest of relevant scholars and industries. In this paper, we propose a new solution to the problem of user selection in mobile crowdsourcing system. The existing user selection schemes mainly include: (1) find a subset of users to maximize crowdsourcing quality under a given budget constraint; (2) find a subset of users to minimize cost while meeting minimum crowdsourcing quality requirement. However, these solutions have deficiencies in selecting users to maximize the quality of service of the task and minimize costs. Inspired by the marginalism principle in economics, we wish to select a new user only when the marginal gain of the newly joined user is higher than the cost of payment and the marginal cost associated with integration. We modeled the scheme as a marginalism problem of mobile crowdsourcing user selection (MCUS-marginalism). We rigorously prove the MCUS-marginalism problem to be NP-hard, and propose a greedy random adaptive procedure with annealing randomness (GRASP-AR) to achieve maximize the gain and minimize the cost of the task. The effectiveness and efficiency of our proposed approaches are clearly verified by a large scale of experimental evaluations on both real-world and synthetic data sets.

**Keywords:** mobile crowdsourcing; user selection; marginalism principle; GRASP-AR

## 1. Introduction

With the development of mobile networks and smart terminals, mobile crowdsourcing [1,2] has gradually evolved into a novel distributed problem-solving paradigm, which uses popular mobile users to collect and process data beyond the past possible range and has become an important research hotspot. The proliferation of smartphones makes mobile crowdsensing applications possible. Newzoo's "2018 Global Mobile Markets Report" [3] shows that the number of global smartphone users has exceeded 3.3 billion so far. As mobile data and hardware become cheaper in the next few years, the number of smartphone users is expected to reach 3.8 billion by 2021, which means there are a large number of potential users in mobile crowdsourcing applications. Furthermore, smartphones are equipped with a variety of powerful built-in sensors, such as GPS, camera, accelerometer, microphone, etc., which enable crowdsourcing users to easily collect basic data of various applications and send sensing data to crowdsourcing platforms.

Inspired by the current popular smart city testbeds [4,5], a typical mobile crowdsourcing platform was proposed. As shown in Figure 1, the mobile crowdsourcing system consists of a large group of crowdsourcing participants distributed in a specific area and a crowdsourcing platform in the cloud, which is connected by mobile network or WiFi. Crowdsourcing participants(a.k.a, Users) use smart devices for road condition monitoring [6], pollution monitoring [7], location services [8], natural disaster assessment [9], etc., and send sensing data to crowdsourcing platforms. However, users consume their resources in performing tasks, such as batteries and computing power. Due to the diversity of individuals and smart devices, the cost of performing tasks is not the same. So they need to get different rewards from crowdsourcing platforms to continue their cooperation. Crowdsourcing platform provides a centralized management platform for sensing data receipt, integration, cleaning, analysis, and further building applications of interest, such as data mining, visualization and data trading, etc. In addition, the crowdsourcing platform has the right to publish tasks and select users.

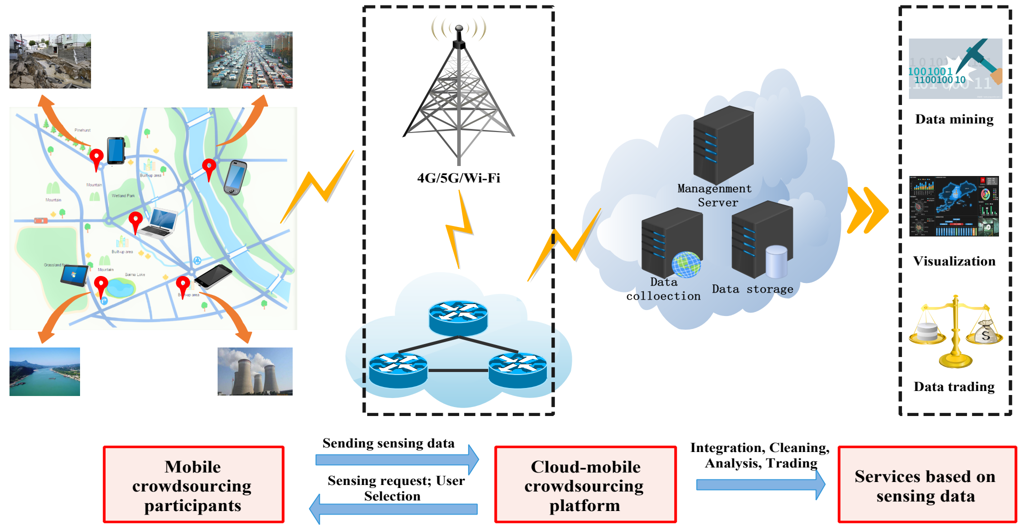

4G/5G/Wi-Fi
Management Server
Data colloection
Data storage
Data mining
Visualization
Data trading
Sending sensing data
Sensing request; User Selection
Integration, Cleaning, Analysis, Trading
Mobile crowdsourcing participants
Cloud-mobile crowdsourcing platform
Services based on sensing data

**Figure 1.** A mobile crowdsourcing system.

To maximize crowdsourcing quality, the crowdsourcing platform needs to select the best subset of users available by matching the task requirements to the user's profile, which is not easy. Crowdsourcing quality is affected by many factors. In addition to spatial location, existing research [10,11] has shown that user reputation, credibility, and time required to complete a perceived task have a significant impact on the quality of crowdsourcing. However, meeting the multi-objective constraints of mobile crowdsourcing remains a challenge. We need to consider not only the distance between the user and the task, but also the completion time of the task, the reputation and credibility of the users. In addition, under the above constraints, how to further select suitable users for the existing mobile crowdsourcing tasks to maximize the quality of crowdsourcing and minimize the incentive budget (i.e., minimum the total incentive cost of the selected users) is a major challenge for the crowdsourcing platform, which is called the mobile crowdsourcing user selection (MCUS) problem.

In truth, these two objectives are contradictory. If we improve one objective, the other will be reduced accordingly. As shown in literature [12], there is no optimal solution that can improve multi-objectives at the same time, but a set of pareto optimal solutions. To solve the problem, there are two standard approaches to formalize the problem: one is to find a subset of users that maximizes the crowdsourcing quality under a given budget; the other is to find a subset of users that minimizes the budget while meeting the minimum crowdsourcing quality requirements.

However, neither of the above methods is ideal in our context, because they all have an artificial predefined condition. Until the available users are selected, the mobile crowdsourcing platform often does not know how well the published tasks match the users. Therefore, it is unrealistic to require the crowdsourcing platform to predetermine a reasonable boundary to achieve a satisfactory compromise between multi-objective. The following two examples illustrate the limitations of both approaches.

**Example 1.** *For simplicity, two potential users are shown in Table 1a, where Quality is the quality of service for users and Cost is the compensation for users. Assume that the upper limit of the budget cost of the crowdsourcing platform is pre-specified at $10. Not surprisingly, User X will be selected and not further inspect User Y. In fact, we can increase the cost slightly to get a more substantial profit. In other words, we increase the cost by $1, but we can get 80% of the service quality (improved by 30%). Arguably, it is worth spending some extra resources.*

**Example 2.** *Similarly, two other users are shown in Table 1b. Assuming that the lower bound of the crowdsourcing quality is predetermined to be 80%, then User A will be selected and User B will not be further examined. In fact, achieving a cost reduction of $6 by slightly relaxing the requirements for crowdsourcing quality (i.e., 2%) is significantly cost effective.*

**Table 1.** Two examples.

| **(a) Example 1** | | | **(b) Example 2** | | |
|---|---|---|---|---|---|
| **User** | **Quality** | **Cost** | **User** | **Quality** | **Cost** |
| X | 50% | $10 | A | 80% | $11 |
| Y | 80% | $11 | B | 78% | $5 |

In practice, both of these scenarios are inevitable because we do not know the distribution of the different solutions and may miss a more desirable solution. Therefore, we need to find all the marginal points before we stop investigation. In other words, we will provide all the pareto-optimal solutions for the crowdsourcing platform to make a reasonable decision.

As mentioned earlier, we can't use the previous methods to solve the user selection problem in mobile crowdsourcing, which is provably a NP-hard problem. To address this challenge, we have made the following significant contributions.

- We propose a scheme inspired by the principle of marginalism in microeconomics, and the problem of user selection in mobile crowdsourcing is formally defined in Section 3. Assuming that the same unit can be used to measure gain and cost in a crowdsourcing platform, we wish to stop selecting new users when the marginal gain is lower than the marginal cost. The marginal gain here refers to the difference between the benefit after and before the selection of users.
- In Section 4, we propose various gain-cost models driven by the Quality of Service (QoS) , which provides a basis for evaluating the value of users' contribution.
- We prove that user selection problem is NP-hard in Section 4.1. Then we propose a greedy random adaptive procedure with annealing randomness (GRASP-AR) to solve the user selection problem in Section 4.2.
- We conduct extensive experiments using real-world and synthetic data sets to evaluate our proposed algorithms on large-scale environment. The results show the effectiveness and efficiency of our proposed approaches in Section 5.

In addtion, we review the previous work on mobile crowdsourcing in Section 2. Section 6 concludes the paper and lay out a research agenda.

## 2. Related Work

In recent years, various researches on mobile crowdsourcing data management have focused on the following five core issues: task assignment [13,14], user selection [15], quality control [16], incentive mechanism [17–19], and privacy protection [20,21]. This article focuses on how to select a suitable group of users to accomplish a specific task. Therefore, the relevant research progress of the latter three issues is not covered in this section.

### *2.1. Task Assignment*

In [22], they consider scheduling different sensing tasks assigned to smartphones in order to minimize the sensing energy consumption while ensuring the Quality of Sensing (QoSS). They proposed an integer linear programming (ILP) formula and two effective polynomial time heuristic algorithms for the corresponding minimum energy multi-sensor task scheduling (MEMS) problem, which has been well evaluated by extensive experimental evaluation. Zhao et al. [23] studied a target-aware task assignment problem that determines the optimal strategy for assigning each task to the right user to maximize the total number of completed tasks, and all users can reach their destination before the deadline after completing the task. To solve the problem, they used tree decomposition technology to separate the users into independent clusters and developed an efficient depth-first search algorithm. To et al. [24] also studied the issue of maximum task assignment. They use the spatial attributes of the problem, namely spatial distribution and user travel costs, to propose the minimum location entropy priority and close distance priority strategies to address these challenges. Wu et al. [25] proposed a real-time, budget-aware spatial crowdsourcing task allocation (RB-TPSC) model, with the goal of increasing task allocation rates and maximizing the expected outcome quality of the staff under a limited budget. The proposed RB-TPSC model can automatically make decisions on task allocation. In addition, Miao et al. [10] proposed a budget-aware task allocation method for spatial crowdsourcing, which is designed to help crowdsourcing platforms make task assignment decisions. Hassan et al. [26] focus on the problem of dynamic task assignment, a distance reliability ratio (DRR) algorithm based on combined fractional programming is proposed to maximize the reliability of the task and minimize the travel cost. DRR maximizes reliability and reduces travel costs by 80% compared to existing algorithms.

### *2.2. User Selection*

Chen et al. [27] focus on the problem of how to choose suitable users to monitor environment by a crowdsensing network, while the total rewards for all selected users is not larger than the limited budget. To solve the problem, they first divide a big critical region into smaller regions of different size, and select some sampling points in the smaller region. Then, they designed a greedy algorithm to select users to cover the maximum sampling points while the total reward does not exceed the limited budget. In addition to time and space factors, Wang et al. [28] also included the data attributes into the mobile crowdsourcing, proposed a method of using the data attributes in the mobile crowdsourcing to select users, and then use the greedy algorithm to select the right users group. Finally, the effectiveness of the proposed method is proved by a large number of experiments on real data. He et al. [29] proposed a new user selection scheme, and designed a genetic algorithm based on greedy approximation and prediction trajectory to solve the problem that the existing user selection strategy does not perform well in vehicle-based group perception. For the multi-user multi-task assignment problem, Abououf et al. [11] proposed a group-based multi-task user selection model, which aim to assign multi-task to users, while maximizing the QoS of tasks and minimizing their completion time. Then the genetic algorithm and the tabu search algorithm were used to complete the user selection problem.

The focus of the above work is to optimize some of the performance for the user by reasonably assigning tasks. Some of them try to match a task to a suitable set of users (task assignment), while others attempt to prioritize users by considering their contributions (user selection). While most studies presuppose artificial constraints (e.g., budget constraint). As we described in the introduction, mobile crowdsourcing platforms are often unclear about the degree of match between published tasks and users. It is unrealistic to require crowdsourcing platform to predetermine a reasonable boundary to achieve a satisfactory compromise between multiobjective. Therefore, we need to redefine the issue of user selection.

## 3. User Selection Driven by the Gain-Cost Models

### *3.1. Problem Definition*

In this section, we will formalize the definition of MCUS problem. For the readers' convenience, the main notations used in this article are listed in Table 2.

**Table 2.** Main notations used throughout the paper.

| Notation | Description |
|---|---|
| $T_i$ | A task published by crowdsourcing platform |
| $L^{T_i}$ | The location of a task $T_i$ |
| $R^{T_i}$ | The minimum reputation requirement of task $T_i$ |
| $Q^{T_i}$ | The minimum quality requirement of task $T_i$ |
| $RC^{T_i}$ | The maximum range constraint of task $T_i$ |
| $U_i$ | A mobile crowdsourcing task participant |
| $L^{U_i}$ | The location of user $U_i$ |
| $R^{U_i}$ | The reputation of user $U_i$ |
| $D^{U_i}$ | The maximum traveling distance constraint set by user $U_i$ |
| $C^{U_i}$ | The cost of user $U_i$ to complete the task |
| $Gain$ | The profit per task |
| $Cost$ | The total cost per task |
| $WTP$ | The willingness to participate of the user |
| $QoS^{U_i}$ | QoS provided by user $U_i$ |
| $QoS^{total}$ | QoS provided by a group of users |

**Definition 1.** *(Multi-objective crowdsourcing task.) Let $T = \{T_1, ..., T_i, ..., T_m\}$, which represents m sets of multi-objective tasks, and each task is defined as a tuple form. $T =< L^{T_i}, R^{T_i}, Q^{T_i}, RC^{T_i} >$, where $L^{T_i}$ is the current location of the task, which can be determined by latitude and longitude. $R^{T_i}$ is the minimum reputation requirement of the task, $Q^{T_i}$ is the minimum quality requirement of the task, and $RC^{T_i}$ is the range constraint of the task.*

**Definition 2.** *(Crowdsourcing participant.) Crowdsourcing participants represent users involved in crowdsourcing task. Let $U = \{U_1, ..., U_i, ..., U_n\}$, which represents a set of n potential users, and each user is defined as a tuple form. $U_i =< L^{U_i}, R^{U_i}, D^{U_i}, C^{U_i} >$, where $L^{U_i}$ represents the current location of the user, which can also be determined by latitude and longitude, and $R^{U_i}$ is the user's reputation score. $D^{U_i}$ is the maximum traveling distance, which is determined by the user. $C^{U_i}$ represents the cost of the user to complete the task, and is also the compensation paid by the platform to the user.*

Reputation score can be obtained through the historical data of user in the crowdsourcing task. A high reputation means that QoS of users is higher. For example, it provides more correct results, takes less time to complete tasks, and so on. In addition, high quality of service means that users need to be paid more, which is in line with our intuition. The detailed calculations will be shown in Section 4.

**Definition 3.** *(Gain for each task.) Gain is determined by the quality of service(QoS) provided by the user, which is denoted by $G(Q)$. The gain satisfies the principle of diminishing marginal benefit [30] in microeconomics (i.e., increasing resources will gradually reduce the unit rate of benefit).*

**Definition 4.** *(Cost of each task.) To obtain the service provided by the user for the task $T_i$, the crowdsourcing platform needs to pay for the selected users, which is also the cost that the crowdsourcing platform must pay for the service. The cost depends on the QoS of each user, so the cost can be expressed by $Cost = \sum_i^{|u|} C(U_i(Q))$, where $|u|$ is the number of users selected.*

Ideally, crowdsourcing platforms wish to maximize gain while minimizing cost. However, achieving both goals is often impossible. The traditional approach is to set constraints on one goal while optimizing on the other. Therefore, we can define the following two optimization problems with constraints.

**Definition 5.** *(The MCUS_$C_1$ problem.) Let U be a set of users, and $\delta_c$ be a budget on cost. The MMCUS_$C_1$ problem finds a subset $\overline{U} \subseteq U$ that maximizes G($\overline{U}$) under constraint $C(\overline{U}) \leq \delta_c$.*

**Definition 6.** *(The MCUS_$C_2$ problem.) Let U be a set of users, and $\delta_g$ be a minimal requirement of gain. The MMCUS_$C_2$ problem finds a subset $\overline{U} \subseteq U$ that minimizes C($\overline{U}$) under constraint $G(\overline{U}) \geq \delta_g$.*

As shown in Examples 1 and 2, neither of these constrained optimization goals is ideal. Inspired by the principle of marginalism [31] in microeconomics, we wish to stop selecting new users when the marginal gain is lower than the marginal cost, which is in line with the economic interests of the crowdsourcing platform. Therefore, the crowdsourcing platform needs to find a set of users with the largest profit (i.e., gain-cost). Assuming that both gain and cost can be measured in the same unit, such as dollars. Additionally, crowdsourcing platform can also apply a budget constraint, but unlike $MMCUS\_C_1$, balancing gain and cost don't require budget constraint. Therefore, we define another goal that user selection.

**Definition 7.** *(The MCUS Marginalism problem.) Let U be a set of users, $U = \{u_1, u_2, ..., u_n\}$, and $\delta_c$ be a budget on cost. The MCUS Marginalism problem finds a subset $\overline{U} \subseteq U$ that maximizes $G(\overline{U}) - C(\overline{U})$ under constraint $C(\overline{U}) \leq \delta_c$, which can be described as follows:*

$$
\begin{aligned}
&Maximize\ G(\overline{U}) - C(\overline{U}) \\
&s.t. = \begin{cases} C(\overline{U}) = \sum_{i=1}^{|\overline{u}|} c_i = \sum_{i=1}^{n} c_i x_i \leq \delta_c \\ x_i = 0\ or\ 1, i = 1, 2, \ldots, n \\ |\overline{u}| \leq n. \end{cases}
\end{aligned} \tag{1}
$$

*where n is the total number of users in a mobile crowdsourcing system, $x_i$ denotes a binary variable: set to 1 if a user is selected and 0 if not. The constraint condition is optional, because we can provide the pareto optimal solution that meets the goal for the platform to make reasonable decisions according to its own constraints.*

### 3.2. Quality of Service (QoS) for Users

In mobile crowdsourcing, the quality of service (QoS) provided by users directly determines whether tasks can be completed. Relevant literature [10,11] show that users' reputation and willingness to participate(WTP) contribute to the evaluation of QoS. When any one of these factors is reduced, the QoS of users will also be reduced, and vice versa. For instance, the higher the reputation, the more favorable the user received in the previous task, the higher the QoS, so the higher the probability of completing the current task. In addition, the quality of crowdsourcing is also determined by the WTP [32] of the user. The higher the WTP, the more human resources the crowdsourcing platform can obtain, which expands the range of users selected and thus increases access to first-rate services. However, the lower the WTP of users, the opposite. Therefore, we consider the reputation(the situation before the task was completed) and the willingness to participate of the users and model the product of these two factors as the standard for QoS in Equation (2).

$$QoS^{U_i} = R^{U_i} \times WTP^{U_i} \in (0, 1) \tag{2}$$

where $R$ is the reputation of the users. Various models [33,34] have been proposed to evaluate the reputation. For simplicity, the popular beta reputation system (RBS) [35] is adopted. However, due to the limitations of RBS, the model does not work well when there are malicious users. For example, savvy users always try any solution that offers more benefits, and they may deliberately report unconfirmed information to obtain unreasonable rewards. And their reputation has not been significantly affected by once or twice unreasonable reporting. Without a penalty for dishonest reporting, crowdsourcing platforms will not only lose access to high-quality information, but also suffer serious financial losses. To avoid this situation, we have improved the RBS and calculated as follow:

$$R = \frac{T+1}{T+F+2} * \xi \; (0 \leq R \leq 1, \; T > 0, \; F > 0) \tag{3}$$

where $T$ and $F$ are the historical numbers of "correct" and "incorrect" results obtained by users when completing the crowdsourcing task. In addition, $\xi$ is a weight associated with malicious events, which was calculated as follows:

$$\begin{cases} if \;\; 0 \leq k < K \;\; then \;\; \xi = \lambda^k \\ if \;\; k \geq K \;\; then \;\; \xi = 0 \end{cases} \tag{4}$$

where $k$ is the malicious event of some user in the crowdsourcing task. $K$ is the threshold for malicious events.

In addition to reputation, the QoS will also be affected by the willingness to participate of users. For example, user $A$ who is far from the task location should have a lower willingness to participate than user $B$ who is closer to the task location, because the distance will increase the cost of usrs. To reduce costs, crowdsourcing platform is also not worth recruiting users from far away. Based on the above analysis, we model the user's willingness to participate and traveling distance as the following equation:

$$WTP = 1 - max[0, min[log_r(D(L^{U_i}, L^{T_i})), 1]] \tag{5}$$

where $D(L^{U_i}, L^{T_i})$ is calculated as the Euclidean distance [36] between the user coordinates and the task coordinates. $r$ is the range constraint of the task. As described in the literature [10], the crowdsourcing task is usually a micro-task, and users are not willing to participate in tasks that are too far away from them. Therefore, the crowdsourcing platform will provide the maximum range constraint of the task. Thus, $WTP$ is a value between 0 and 1. Within the maximum range constraint of the task, if the user is closer to the task location, then $WTP$ is closer to 1. If the location of the user is beyond the maximum range constraint of the task, then $WTP$ = 0.

Based on the above analysis, substitute ((3)–(5)) into (2), then we can evaluate the $QoS^{U_i}$ of each user, and a task is completed by the cooperation of multiple users. We can increase the number of users until the marginal cost exceeds the marginal gain for the next incremental user. In other words, MCUS Marginalism problem aims to maximize gain of each task and ensure that a group of users is selected if adding a new user will increase the probability of the task success. This probability is defined as the number of correct reports submitted.

After the calculation of $QoS^{U_i}$, the probability of at least one success of the task is calculated as the binomial distribution of the total $QoS$ obtained by the task. The $QoS^{total}$ of the selected a group of users is estimated as follow:

$$QoS^{total} = 1 - \prod_{i=1}^{|u|}(1 - QoS^{U_i}) \tag{6}$$

## *3.3. QoS-Based Gain-Cost Models*

In this paper, we consider the effect of different gain-cost models [37,38] on user selection, take the $QoS$ of user in Section 3.2 as an important gain factor, and establish three gain models.

- LinearGain assumes that the gain grows linearly with the QoS of users (For writing convenience, record it as $Q(u)$) and set $G(u) = 100Q(u)$
- QuadGain assumes that the gain increases quadratically with $Q(u)$ and set $G(u) = 100Q^2(u)$
- StepGain assumes that reaching a milestone of $Q(u)$ will significantly increase gain and set:

$$G(u) = \begin{cases} 100Q(u) & : 0 \le Q(u) < 0.2 \\ 100 + 100(Q(u) - 0.2) & : 0.2 \le Q(u) < 0.5 \\ 150 + 100(Q(u) - 0.5) & : 0.5 \le Q(u) < 0.8 \\ 200 + 100(Q(u) - 0.8) & : 0.8 \le Q(u) < 1 \end{cases}$$

We assign the cost of a user in [10, 40] in four ways:

- RandomCost assigns a random integer cost in [10, 40];
- LinearCost assumes that the cost grows linearly with $Q(u)$ and set $C(u) = 30Q(u) + 10$;
- QuadCost assumes that the cost increases quadratically with $Q(u)$ and set $C(u) = 30Q^2(u) + 10$
- StepCost assumes that reaching some milestone of $Q(u)$ will significantly increase cost and set:

$$C(u) = \begin{cases} 10 & : 0 \le Q(u) < 0.2 \\ 20 & : 0.2 \le Q(u) < 0.5 \\ 30 & : 0.5 \le Q(u) < 0.8 \\ 40 & : 0.8 \le Q(u) \le 1 \end{cases}$$

## 4. Optimization Algorithm

In this section, we first analyze the complexity of solving MCUS Marginalism problem, and we rigorously prove that MCUS Marginalism problem is a NP-hard problem. Then we propose a greedy random adaptive procedure with annealing randomness(GRASP-AR), which attempt to overcome the limitations of deterministic algorithms (based on adding what is apparently the best element to the partial solution), that is, our algorithms strive to ensure that the solution is globally optimal because they explore the search space in a comprehensive way.

### *4.1. Complexity Analysis of MCUS Marginalism Problem*

It is very important to solve the MCUS Marginalism problem with an efficient algorithm. Unfortunately, as we will prove next, MCUS Marginalism problem is a NP-hard.

**Theorem 1.** *MCUS Marginalism problem defined in Defintion 7 is a NP-hard Problem.*

**Proof.** The *MCUS* Marginalism problem can be proved by reducing $MCUS\_C_1$ problem. Further, we can prove that the decision version of $MCUS\_C_1$ is NP-complete. Next, we should find a known NPC problem and then try to reduce it.

We use the knapsack problem as a known NP complete problem. The knapsack problem is defined as follows. For an instance A of knapsack problem and a set of object $K = \{k_1, k_2, ..., k_n\}$ with value and weight , where $k_i$ is represented by $v_i$ and $w_i$ respectively. The question is whether exists a set $\overline{K} \subseteq K$ that maximizes the value of members from $\overline{K}$ (i.e., $v_{max}(\overline{K})$), and further $\sum_{i=1}^{|\overline{K}|} w_i \le \eta$, where $\eta$ is upper bound of the capability.

Next, we change instance $A$ to an instance of $MCUS\_C_1$. We construct a user instance $B$ with cost upper bound $\delta$, and represent it as $U = \{u_1, u_2, ..., u_n\}$. For each $u_i$, its gain and cost are $g_i$ and $c_i$, respectively. The $MCUS\_C_1$ problem is to find a set $\overline{U} \subseteq U$ to maximize the gain from the members of $\overline{U}$ (i.e., $G_{max}(\overline{U})$), and further $\sum_{i=1}^{|\overline{U}|} c_i \le \delta$. We assume that $\overline{K}$ can be used as a solution for instance $A$. By trying to select $|\overline{U}|$ users in the set $U$, the formed $\overline{U}$ can be used as a solution for $MMCUS\_C_1$.

With the construction approach of the solution, $\sum_{i=1}^{|\overline{U}|} c_i \leq \delta$ and a maximal $g_{max}(\overline{U})$ imply $\sum_{i=1}^{|\overline{K}|} u_i \leq \eta$ and a maximal $v_{max}(\overline{K})$, respectively.

Then, we can simply see that the reduction from A to B ends in polynomial time, since the knapsack problem is a NP-hard problem, so the $MCUS\_C_1$ is NP-hard. While $MCUS$ Marginalism problem can be reduced to $MCUS\_C_1$, so $MCUS$ Marginalism problem is also a NP-hard problem. $\square$

### *4.2. GRASP with Annealing Randomness (GRASP-AR)*

In the previous section, we have shown that MCUS marginalism problem is a NP-hard problem. As the scale of users continues to expand, solving MCUS Marginalism problem is limited by both memory and time. In practice, the number of decision variables increases exponentially as the scale of users increases. Even if computing resources can increase indefinitely, the exact solution may not be found in a reasonable amount of time. To address the problem, we proposed a greedy randomized adaptive search procedure with annealing randomness as a trade-off between computation time and quality of found solutions.

GRASP [39,40] is a multi-start meta-heuristic algorithm, which consists of two phases: construction phase and local search phase. In the construction phase, the iterative constructs a feasible solution, one element at a time. The greedy randomized algorithm is first used to select the top-*k* candidates from the generated profits, and the best solution is selected from multiple iterations. The algorithm is adaptive and provides a good initial solution while maintaining a certain degree of diversification to avoid convergence toward local optima. In the local search phase, we introduce a simulated annealing (SA) [41] meta-heuristic, whose effectiveness depends on the fact that it accepts a non-improved solution within a certain probability. In other words, it can accept a solution that is worse than the best solution found at the time. The probability of this solution depends on the degree to which the new solution differs from the optimal solution and a further parameter, the synthesis temperature excited by the metallurgical annealing process. As the value of the parameter decreases gradually, the randomness of the method also gradually decreases (the lower the value, the lower the randomness). Because SA can search for feasible solutions in a larger range, making it possible for the algorithm to jump out of the local optimal solution and find a global optimal solution.

**Algorithm 1:** GRASP-AR$(U, \lambda, k)$.

```
Input  : $U$: users for selection; $\lambda$: the number of iterations;
         $k$: finding top-$k$ candidates
Output: $\overline{U}$: selected users
1 $\overline{U}_{best} \leftarrow \emptyset$; $P_h \leftarrow \emptyset$; \\ records the highest gain;
2 for $i \leftarrow 1$ to $\lambda$ do
3     $(\overline{U}, G, C) \leftarrow$ BuildSolution$(U, \emptyset, 0, 0, k)$;
4     $(\overline{U}, G, C) \leftarrow$ LocalSearch$(U, \overline{U}, G, C, k)$;
5     if $G - C > P_h$ then
6         $\overline{U}_{best} \leftarrow \overline{U}$; $P_h \leftarrow G - C$;
7 return $\overline{U}_{best}$;
```

Algorithm 1 introduces the main processing phases of GRASP-AR. The algorithm ensures that a subset of users with the largest marginal gain is found. The algorithm performs $\lambda$ iterations. In each iteration, the construction phase constructs an initial solution $\overline{U}$, then the local search phase uses a simulated annealing strategy to further find a viable solution in the $\overline{U}$ neighborhood. Finally, it returns the best solution from all iterations.

The construction process of greedy randomized is given in Algorithm 2. First, a given set of candidate users is initialized and a group of users is added iteratively in a greedy randomized. In each iteration (Step. 2–14), we select the user with the maximum incremental gain from the remaining user set $U \setminus \overline{U}$ (Step. 5). In other words, we check one by one whether the maximum gain achieved by the remaining users can exceed the current best solution, and skips the user if not. Next, Step. 6 evaluates the difference between the marginal gain and marginal cost of the selected users. The top-*k* user sets are selected in this way (Step. 7–10). Finally, Step. 12–15 choose subset of users with the highest gain.

**Algorithm 2:** Procedure BuildSolution($U, \overline{U}, G, C, k$).

```
Input  : $U$: users for selection; $\overline{U}$: already selected users;
         $G$: gain for $\overline{U}$; $C$: cost for $\overline{U}$; $k$: finding top-$k$ candidates
Output : $(\overline{U}_{best}, G, C)$: the newly selected users and their gain and cost
1  $\overline{U}_{best} \leftarrow \overline{U}$; $P_h \leftarrow G - C$; // Initialize the best solution as the input
2  for $i \leftarrow 1$ to $|U| - |\overline{U}|$ do
3      $Optimal \leftarrow \emptyset$; $\mathbb{A} \leftarrow \emptyset$; Store the top-k candidates
4      foreach $u \in U \backslash \overline{U}$ do
5          if $G_{total} - C - c(u) > P_h$ then
6              $P \leftarrow G(Q(\overline{U} \cup u))$ -$C$ -$c(u)$;
7              $\overline{k} \leftarrow$ rank of $P$ in $\mathbb{A}$;
8              if $\overline{k} \leq k$ then
9                  $Optimal \leftarrow Optimal \cup U$;
10                 $\mathbb{A} \leftarrow \mathbb{A} \cup P$;
11                 Update a set of users($Optimal$)
12     // Randomly choose $P_r$ from $\mathbb{A}$
13     if $P_r > P_h$ then
14         $P_h \leftarrow P_r$;
15         $\overline{U}_{best} \leftarrow \overline{U}$;
16 return ($\overline{U}_{best}$, G($\overline{U}_{best}$), C($\overline{U}_{best}$));
```

**Algorithm 3:** Procedure LocalSearch(U, $\overline{U}$, G, C, k).

```
Input  : $U$: users for selection; $\overline{U}$: already selected users;
         G: gain for $\overline{U}$; C: cost for $\overline{U}$; $k$: finding top-$k$ candidates
Output : ($\overline{U}_{best}$, G, C): the newly selected users and their gain and cost
1  while $T > T_{low}$ do
2      foreach $u \in \overline{U}$ do
3          $\overline{U}_0 \leftarrow \overline{U} \backslash u$;
4          $C_0 \leftarrow C - c(u)$;
5          $G_0 \leftarrow G(Q(\overline{U}_0))$;
6          $(\overline{U}_0, G_0, C_0) \leftarrow$ BuildSolution$(U, \overline{U}_0, G_0, C_0, k)$;
7          $\Delta = (G_0 - C_0) - (G - C)$
8          if $\Delta > 0$ then
9              // always accept better solution as current one update the better found
                  solution so far
10             $\overline{U} \leftarrow \overline{U}_0$; $G = G_0$; $C = C_0$;
11         else if $exp(\Delta/\alpha T) > Random(0,1)$ then
12             // accept the worse solution as a current one with probability given by above
                  formula
13             $\overline{U} \leftarrow \overline{U}_0$; $G = G_0$; $C = C_0$;
14         $T = \alpha * T$
15 return $\overline{U}, G, C$;
```

The local search phase takes the initial solution as input and iteratively explores its neighborhood for a better solution. In this work, we introduce SA meta heuristic method. SA is a general probability algorithm. Its starting point is based on the annealing process of solid materials in metallurgical processes. It is initialized with a parameter called temperature, denoted $T$, which accepts a non-improved solution within a certain probability. According to the cooling rate factor $\alpha$, the temperature slowly drops during execution. The lower the temperature, the lower the probability of choosing a worse solution in the next iteration.

The proposed SA is given in Algorithm 3. First, it takes the construction phase of the solution as input, which is the best solution found so far. Next, in each iteration, it compares the current solution with (1) the solution of removing $u$ (2) the candidate solution with the remaining user subset replacement $u$, and represents the difference between them as $\Delta$. If the candidate solution is better, consider it as the current solution and update the best solution found. If the candidate solution is worse, it is accepted as the new current solution with the probability $exp(\Delta/\alpha T)$. Finally, the temperature decreases with the cooling factor until the preset minimum temperature stops.

## 5. Experimental Evaluation

The purpose of this section is to examine the performance of our algorithm using both real-world and synthetic datasets. We compare GRASP-AR with three baseline approaches on a series of mobile crowdsourcing tasks. First, we describe our experimental design, then we analyze the results under various experimental settings.

All experiments were coded in Python under Windows 10 for Education platform on an Intel Core i7 2.8 GHz processor with 16 GB of RAM.

### *5.1. Experiment Design*

#### 5.1.1. Datasets

**Real-world data**: To evaluate the performance of the GRASP-AR algorithm, we used a task assignment data set [42] published by the China Society for Industrial and Applied Mathematics, which contains 835 tasks and 1877 users from four Chinese cities (i.e., Guangzhou, Foshan, Shenzhen and Dongguan). The distribution of task locations and user locations are shown in Figure 2. In the experiment, we set the parameter $RC^{T_i}$ = 50 km. In addition, the dataset also provides the reputation of all users, which is scaled to a range of (0, 1). The dataset parameter settings are summarized in Table 3.

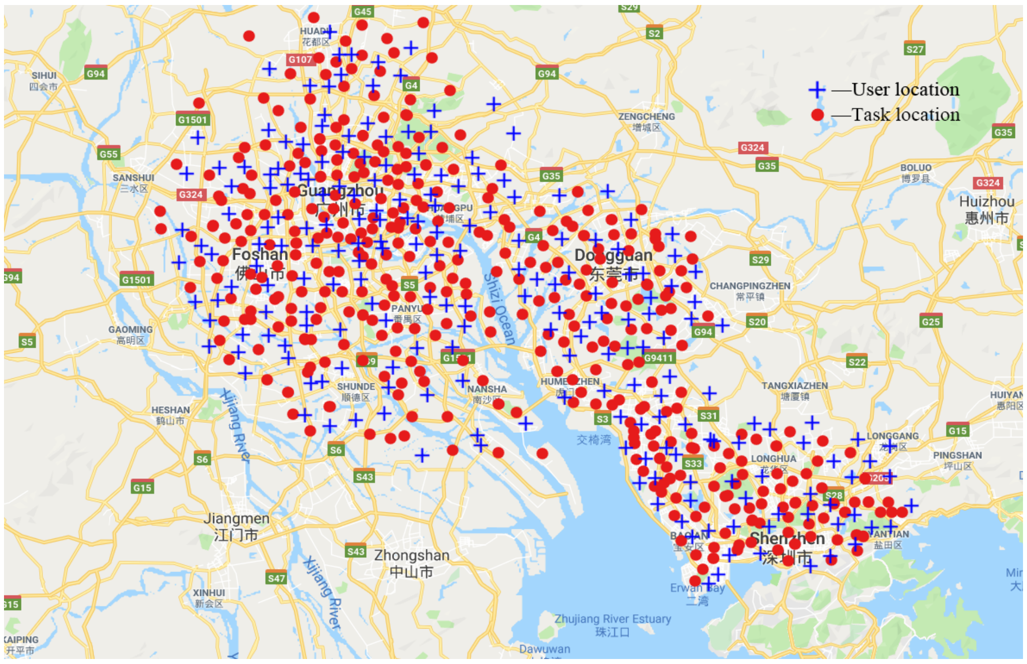


**Figure 2.** A real-world data set of task allocation.

**Synthetic data**: We follow the existing methods [43,44] to generate synthetic data. Assuming that the location of tasks and users are distributed in a 1 km $\times$ 1 km 2D space, then longitude and latitude are uniformly distributed in (0, 1). In addition, each user has a reputation score $R$ that reflects the completion of tasks in the past. For simplicity, the reputation of users is randomly selected from the range of (0, 1). The $n$ users and $m$ tasks are selected from {100, 200, 300, 400, 500, 600, 700, 800, 900} and {100, 200, 300, 400, 500, 600, 800} respectively. We use Euclidean distance to quantify the traveling distance between tasks and users. The synthetic data parameter settings are summarized in Table 4, where default values are highlighted in bold.

**Table 3.** The real-world data parameters and their values.

| Parameter | Description | Value |
|---|---|---|
| $m$ | Number of crowdsourcing tasks | 835 |
| $n$ | Number of users | 1877 |
| $RC^{T_i}$ | Range constraints for each task | 50 km |
| $R$ | Reputation of user | (0, 1) |

**Table 4.** The synthetic data parameters and their values.

| Parameter | Description | Range of Values |
|---|---|---|
| $n$ | Number of users | {100, 200, **300**, 400, 500, 600, 700, 800, 900} |
| $R$ | Range of the reputation of a user | [0, 1] |
| $m$ | Number of crowdsourcing tasks | {100, 200, 300, **400**, 500, 600, 700, 800} |
| $g$ | Grid size | 1 km $\times$ 1 km squares |

### 5.1.2. Baseline Algorithms

To evaluate our GRASP-AR algorithm, we introduce three other algorithms. One is Particle Swam Optimization (PSO) algorithm as in [45]. PSO is an optimization method based on Swarm Intelligence, which originated from the research on predation behavior of birds. Compared with other modern optimization methods, PSO is featured by few parameters to be adjusted, simple operation and fast convergence, which has become a hotspot in the field of modern optimization methods. We have set the size of the swarm to 30, the inertia weight linearly varied between 0.9 to 0.4, and the acceleration coefficients to 2. Also, we compare our GRASP-AR algorithm with its two simplified versions, namely Greedy Randomized Algorithm (GRA) and Simulated Annealing (SA). The former one invokes *BuildSolution* with $k$ = 1 (see Algorithm 2). The latter one essentially invokes *LocalSolution* (see Algorithm 3). The difference is that the initial solution for SA is generated randomly.

Additionally, we adjust parameters separately and incrementally, i.e., when a parameter is tuned, others are set to the default values. While next parameters are investigated, previous ones are taking the best value found so far. Table 5 lists the range of parameter values and default value for GRASP-AR algorithm.

**Table 5.** The GRASP-AR algorithm parameters and their values.

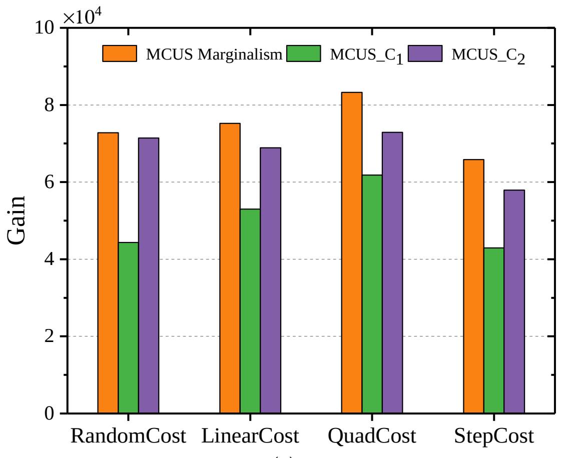
| Parameter | Description | Range of Values | Default Value |
|---|---|---|---|
| $k$ | Top-k candidates | {5, 15, 30, 50, 75} | 15 |
| $\lambda$ | Number of repetitions | {10, 30, 60, 100, 150, 210} | 150 |
| $\alpha$ | Cooling rete factor | {0.990, 0.995, 0.999, 0.9995, 0.9999} | 0.995 |
| $T$ | Initial temperature | {200, 400, 600, 800, 1000} | 400 |
| $T_{low}$ | Termination temperature | 0.1 | 0.1 |

### *5.2. Evaluation Results*

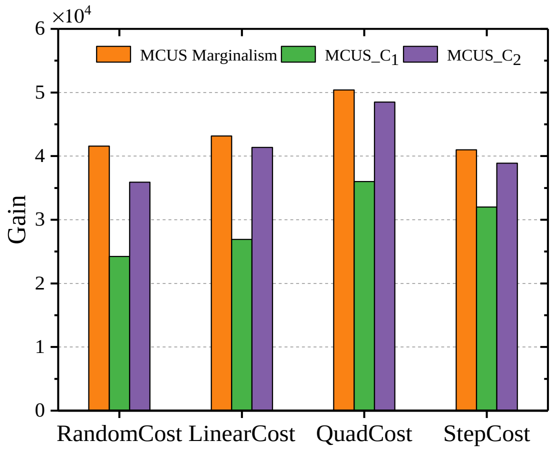
#### 5.2.1. Comparison of Three Selection Schemes under Different Gain-Cost Models Using Real Dataset

We considered three user selection schemes mentioned in Section 3.1: (1) $MCUS\_C_1$ with $\delta_c = \frac{G(1)}{2}$ (G(1) corresponds to the maximum gain), (2) $MCUS\_C_2$ with $\delta_g = G(0.8)$, and (3) $MCUS\ Marginalism$ with $\delta_c = \infty$. We use the default experimental parameters in Table 3 for each scheme.

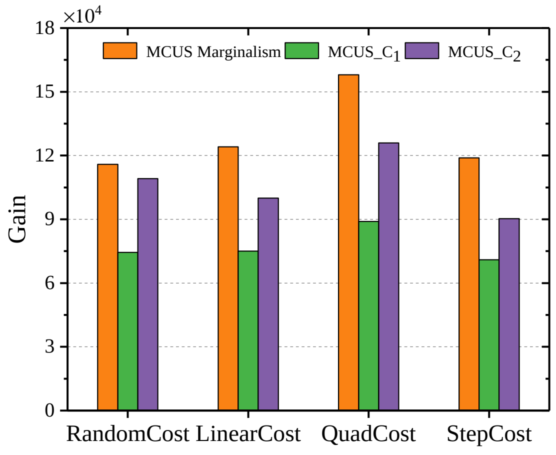
To compare the effects of different gain-cost models on user selection, we used LinearGain, QuadGain, StepGain and various cost models on the real-world dataset. As shown in Figure 3, different gain models show the same patterns. Specifically, the user selection scheme ($MCUS\ Marginalism$) we proposed achieved the highest gain in most cases. Taking Figure 3a as an example to illustrate, $MCUS\ Marginalism$ is nearly 32% higher than $MCUS\_C_1$ and 9.6% higher than $MCUS\_C_2$ in terms of average gain. This is because $MCUS\_C_1$ costs a lot, and $MCUS\_C_2$ always stops at a fairly low gain. This difference is more obvious in the StepGain model.



(a)



(b)



(c)

**Figure 3.** Comparison of three selection schemes. (**a**) LinearGain; (**b**) QuadGain; (**c**) StepGain.

#### 5.2.2. Performance Comparison of Four Algorithms under Different Gain-Cost Models Using Real Dataset

We used real-world dataset and generated a total of six instances based on different gain-cost models. GRASP-AR, PSO, GRA, SA were used to obtain the best value, the worst value, the mean value, and the standard deviation (S.D) for solving each instance 20 times independently.

Table 6 shows the results of RandomCost and LinearCost. Firstly, in terms of the results of best value, we observed that GRASP-AR achieved 5 best results in 6 instances, SA achieved the best results in only one instance, while others did not. Secondly, the performance of various StepGain models is poor, which may be due to the discontinuity of the gain with the increase of QoS.

**Table 6.** Performance Comparison of GRASP-AR, PSO, GRA and SA.

| Index | Gain | Cost | Algorithm | Best | Mean | Worst | S.D |
|---|---|---|---|---|---|---|---|
| 1 | Linear | Random | GRASP-AR | 72,768 | 71,395 | 68,742 | 112.56 |
| | | | PSO | 69,128 | 63,746 | 55,221 | 437.24 |
| | | | GRA | 70,153 | 66,712 | 58.968 | 314.78 |
| | | | SA | 70,994 | 67,475 | 61,827 | 294.28 |
| 2 | | Linear | GRASP-AR | 75,245 | 72,847 | 70,328 | 146.15 |
| | | | PSO | 71,824 | 64,723 | 58,863 | 398.16 |
| | | | GRA | 73,080 | 68,451 | 62,058 | 295.14 |
| | | | SA | 72,993 | 69,418 | 64,862 | 278.35 |
| 3 | Quad | Random | GRASP-AR | 41,582 | 38,358 | 35,210 | 125.85 |
| | | | PSO | 38,692 | 34,917 | 31,026 | 358.14 |
| | | | GRA | 39,822 | 37,132 | 34,328 | 158.86 |
| | | | SA | 40,217 | 36,258 | 33,461 | 198.53 |
| 4 | | Linear | GRASP-AR | 43,158 | 41,062 | 38,894 | 140.85 |
| | | | PSO | 40,558 | 38,258 | 34,016 | 342.10 |
| | | | GRA | 41,139 | 38,153 | 35,927 | 194.85 |
| | | | SA | 41,056 | 36,534 | 33,635 | 221.52 |
| 5 | Step | Random | GRASP-AR | 115,816 | 105,848 | 97,451 | 296.43 |
| | | | PSO | 104,487 | 88,945 | 80,032 | 578.82 |
| | | | GRA | 109,922 | 92,628 | 85,396 | 372.52 |
| | | | SA | 116,132 | 99,863 | 84,150 | 606.21 |
| 6 | | Linear | GRASP-AR | 124,136 | 116,980 | 109,628 | 152.18 |
| | | | PSO | 105,662 | 96,463 | 86,916 | 428.47 |
| | | | GRA | 115,132 | 103,558 | 90,284 | 265.37 |
| | | | SA | 113,472 | 100,824 | 88,153 | 416.85 |

In addition, we used GAP to evaluate the statistical characteristics of the average performance of all algorithms. The calculation of GAP is the same as [38], which is measured by the relative difference between the best value and the average value:

$$GAP = \frac{|best - mean|}{best} \times 100\% \tag{7}$$

Then, we can compare the average performance of all algorithms by GAP fitting curve. The closer the gap fitting curve is to the abscissa axis, the better the average performance of the algorithm. The fitting curves of each algorithm are given in Figure 4. It can be seen that the performance of GRASP-AR is the best among the four algorithms, because the gap fitting curve is the closest to the abscissa axis.

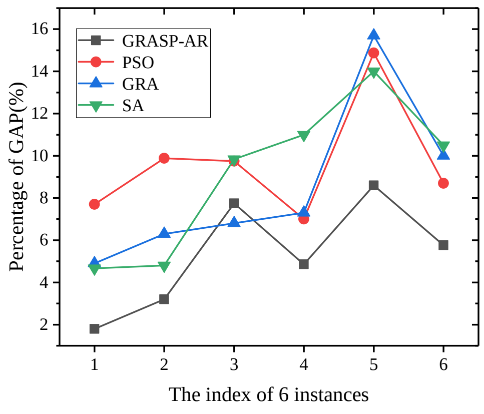


**Figure 4.** GAP fitting curve of real-world dataset under different gain-cost models.

Moreover, we plot the histogram according to S.D and evaluate the stability of the four algorithms by the distribution of the columns. As shown in Figure 5, the stability of GRASP-AR is much higher than the other three algorithms.

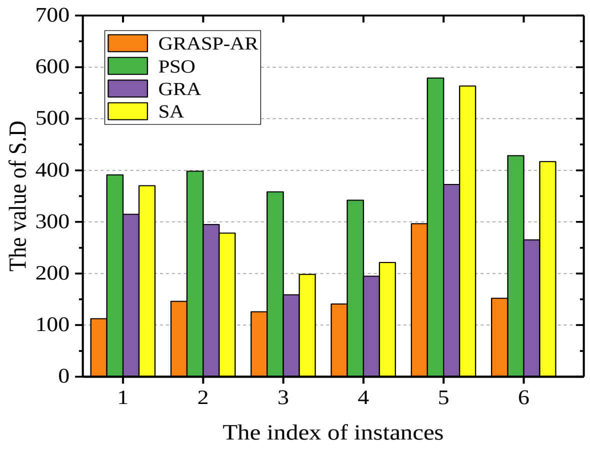


**Figure 5.** Standard deviation (S.D) histogram of real-world dataset under different gain-cost models.

#### 5.2.3. Performance Comparison of Various Parameter Combinations for GRASP-AR Using Synthetic Dataset

We used LinearGain and RandomCost to find the best selection percentage and running time for different combinations of $\lambda$ and $k$. Figure 6 shows the results of our experiment. We have three conclusions. First, there is no doubt that the more iterations, the longer the time, but the more iterations, the better the results. Second, better results are often obtained when $k = 15$. This is because if the value of $k$ is too low, the best solution may not be found, and if the value of $k$ is too high, it is close to random search, which will reduce the quality of the results. Finally, when $k$ increases gradually, the running time increases, while $k$ exceeds 50, the running time decreases. This is because when $k$ is large, it is less likely to find a better solution in the construction phase, so there are fewer iterations in the local search phase.

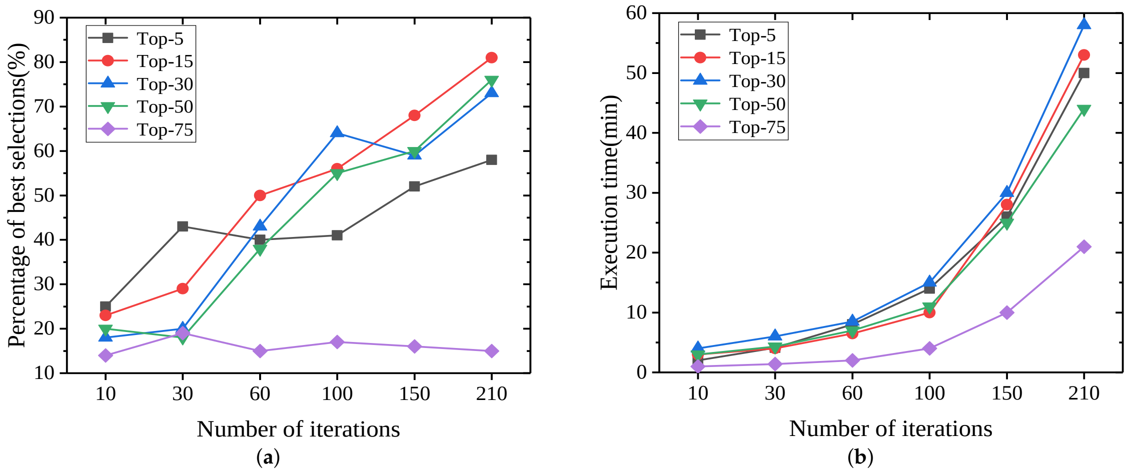


**Figure 6.** Effectiveness and efficiency of various parameter combinations for GRASP-AR. (**a**) Percentage of Best Selections; (**b**) Execution Time.

#### 5.2.4. Performance Comparison of Four Algorithms under Different Number of Tasks Using Synthetic Dataset

The experiment in this section aims to evaluate the performance of the algorithm according to the number and location of different tasks when determining the number and location of users. We consider 200 users and gradually increase the number of tasks from 100 to 1000. Moreover, we use LinearGain and RandomCost models.

As shown in Figure 7a,b, while all algorithms aim to maximize the gain of the task, GRASP-AR achieves the maximum (total) gain of all tasks and the average gain of each task, respectively. Specifically, GRASP-AR performs approximately 32%, 17% and 14% better than PSO and GRA and SA, respectively. It is no surprise that GRASP-AR provides a good starting point and maintains a variety of solutions in the construction phase, which provides a good foundation for the final user selection. Secondly, the local search phase explores the neighborhood of the solution and further improves the solution. Furthermore, we can see that the SA cannot be extended over a large number of tasks, which significantly reduces the task gain.

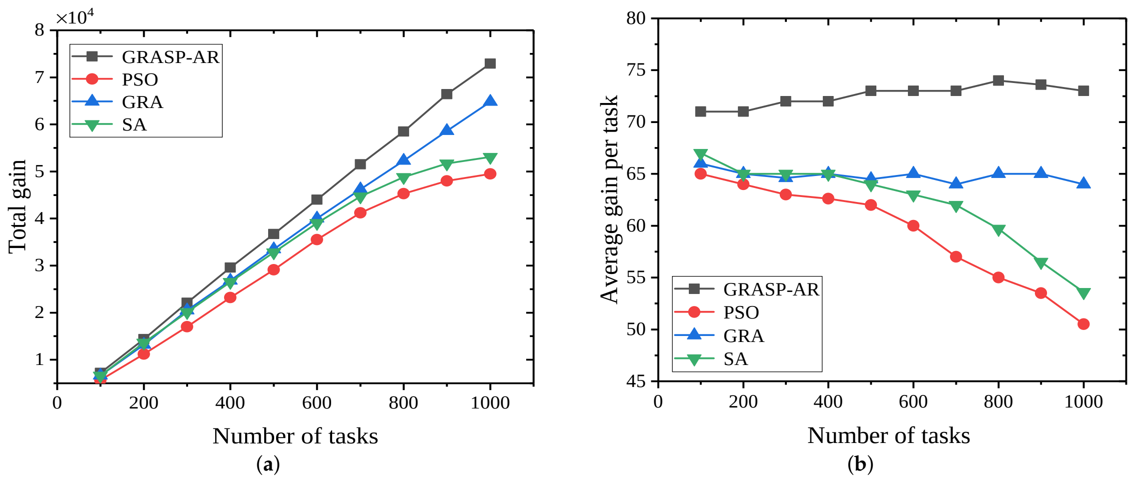


**Figure 7.** Effectiveness and efficiency of various parameter combinations for GRASP-AR. (**a**) The total gain achieved for all tasks vs. the number of tasks available. (**b**) The average gain achieved per task vs. the number of tasks available.

#### 5.2.5. Performance Comparison of Four Algorithms under Different Number of Users Using Synthetic Dataset

The experiment in this section aims to evaluate the performance of various algorithms according to different number of users when determining the number and location of tasks. We considered 50 tasks and gradually increased the number of users from 100 to 800. Similarly, we use the linear gain and random cost models.

Figure 8 shows the effect of available users on the total gain of 50 tasks. It can be seen that with the increase of the number of potential users, the contribution of the four algorithms to the total gain of 50 tasks is increasing, and their fitting curves are getting closer. This is because the more users in the search space, the more potential users to provide high-quality service, and a better subset of users can be found for the task. It is worth noting that the performance of GRASP is still higher than other algorithms. It is 15.8%, 14.3% and 8.7% higher than SA, PSO and GRA, respectively, in terms of average gain.

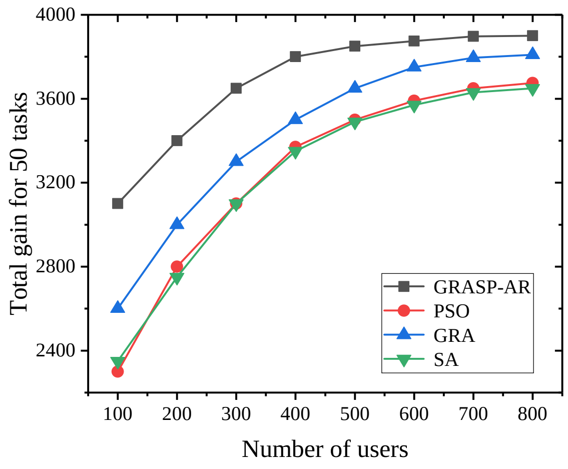


**Figure 8.** The total gain achieved for 50 tasks vs. the number of users available.

## 6. Conclusions and Future Work

This paper studies the problem of user selection in mobile crowdsourcing. We propose a marginalism problem of user selection to achieve maximize the quality of service of task while minimizing the total incentive cost. In addition, in order to estimate the contribution of users to task, we construct various gain-cost models driven by willingness to participate and reputation of the users. To address user selection marginalism problem, we develop a greedy random adaptive procedure with annealing randomness that can efficiently retrieve the potential user selection solution feasible space. Experimental results show the effectiveness of our algorithms on both real-world and synthetic datasets.

There are many opportunities to extend this work for full-fledged user selection for moblie crowdsourcing. We next lay out a research agenda by describing several future research directions.

- User contribution measures: In addition to the user's willingness to participate and reputation, there are a variety of indicators to evaluate the user's contribution to the task, such as credibility, task completion time, etc. Future work includes efficiently estimating user contribution and selecting user given these new measures.
- Improved gain and cost models: When we have multi-dimensional user contribution measures, the gain model can be much more complex. Similarly, the cost model may be more complex based on some specific pricing strategies [46,47]. Future work includes designing more complex gain-cost models and studying their effect on user selection.

**Author Contributions:** Conceived and designed the experiments, J.Y. and X.B; methodology, J.Y. and X.B.; Performed the experiments/Wrote the paper, J.Y.; Supervision, X.B. and C.X.; funding acquisition, X.B. and C.X.

**Funding:** This research was supported in part by National Nature Science Foundation of China (Grant No. 61873299, 61702036, 91646202), National Key Research and Development Program of China (Grant No. 2016YFB1001404, 2018YFB1404400, 2018YFB1402700).

**Acknowledgments:** The authors are deeply thankful to the editor and reviewers for their valuable suggestions to improve the quality of this manuscript.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Guo, B.I.N.; Wang, Z.H.U.; Yu, Z.; Wang, Y.U.; Yen, N.Y.; Huang, R.; Zhou, X. Mobile Crowd Sensing Survey. *ACM Comput. Surv.* **2015**, *48*, 31. [CrossRef]
2. Boubiche, D.E.; Imran, M.; Maqsood, A.; Shoaib, M. Mobile Crowd Sensing—Taxonomy, Applications, Challenges, and Solutions. *Comput. Hum. Behav.* **2018**. [CrossRef]

3. Global Mobile Markets Report. 2018. Available online: https://newzoo.com/insights/trend-reports/newzoo-global-mobile-market-report-2018-light-version/ (accessed on 5 June 2019).
4. Amaxilatis, D.; Mylonas, G.; Diez, L.; Theodoridis, E.; Gutiérrez, V.; Muñoz, L. Managing Pervasive Sensing Campaigns via an Experimentation-as-a-Service Platform for Smart Cities. *Sensors* **2018**, *18*, 2125. [CrossRef]
5. Choque, J.; Diez, L.; Medela, A.; Muñoz, L. Experimentation Management in the Co-Created Smart-City: Incentivization and Citizen Engagement. *Sensors* **2019**, *19*, 411. [CrossRef]
6. Khan, S.Z.; Rahuman, W.M.A.; Dey, S.; Anwar, T.; Kayes, A.S.M. RoadCrowd: An Approach to Road Traffic Forecasting at Junctions Using Crowd-Sourcing and Bayesian Model. In Proceedings of the 2017 International Conference on Research and Innovation in Information Systems (ICRIIS), Langkawi, Malaysia, 16–17 July 2017; pp. 1–6.
7. Yi, L.; Deng, X.; Wang, M.; Ding, D.; Wang, Y. Localized Confident Information Coverage Hole Detection in Internet of Things for Radioactive Pollution Monitoring. *IEEE Access* **2017**, *5*, 18665–18674. [CrossRef]
8. Yuan, N.J.; Zheng, Y.; Zhang, L.; Xie, X. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 2390–2403. [CrossRef]
9. Abu-Elkheir, M.; Hassanein, H.S.; Oteafy, S.M.A. Enhancing Emergency Response Systems through Leveraging Crowdsensing and Heterogeneous Data. In Proceedings of the 2016 International Wireless Communications and Mobile Computing Conference (IWCMC), Paphos, Cyprus, 5–9 September 2016; pp. 188–193.
10. Miao, C.; Yu, H.; Shen, Z.; Leung, C. Balancing Quality and Budget Considerations in Mobile Crowdsourcing. *Decis. Support Syst.* **2016**, *90*, 56–64. [CrossRef]
11. Abououf, M.; Mizouni, R.; Singh, S.; Otrok, H.; Ouali, A. Multi-Worker Multi-Task Selection Framework in Mobile Crowd Sourcing. *J. Netw. Comput. Appl.* **2019**, *130*, 52–62. [CrossRef]
12. Wang, L.; Yu, Z.; Han, Q.; Guo, B.; Xiong, H. Multi-Objective Optimization Based Allocation of Heterogeneous Spatial Crowdsourcing Tasks. *IEEE Trans. Mob. Comput.* **2018**, *17*, 1637–1650. [CrossRef]
13. Wang, J.; Wang, L.; Wang, Y.; Zhang, D.; Kong, L. Task Allocation in Mobile Crowd Sensing: State-of-the-Art and Future Opportunities. *IEEE Internet Things J.* **2018**, *5*, 3747–3757. [CrossRef]
14. Bellavista, P.; Corradi, A.; Foschini, L.; Ianniello, R. Scalable and Cost-Effective Assignment of Mobile Crowdsensing Tasks Based on Profiling Trends and Prediction: The ParticipAct Living Lab Experience. *Sensors* **2015**, *15*, 18613–18640. [CrossRef]
15. Chen, Y.; Lv, P.; Guo, D.; Zhou, T.; Xu, M. Trajectory Segment Selection with Limited Budget in Mobile Crowd Sensing. *Pervasive Mob. Comput.* **2017**, *40*, 123–138. [CrossRef]
16. Wang, S.; Zhao, Y.; Huang, L.; Xu, J.; Hsu, C.H. QoS Prediction for Service Recommendations in Mobile Edge Computing. *J. Parallel Distrib. Comput.* **2017**, *127*, 134–144. [CrossRef]
17. Jaimes, L.G.; Vergara-Laurens, I.J.; Raij, A. A Survey of Incentive Techniques for Mobile Crowd Sensing. *IEEE Internet Things J.* **2015**, *2*, 370–380. [CrossRef]
18. Jin, H.; Su, L.; Chen, D.; Nahrstedt, K.; Xu, J. Quality of Information Aware Incentive Mechanisms for Mobile Crowd Sensing Systems. In Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc'2015, Hangzhou, China, 22–25 June 2015; pp. 167–176.
19. Jin, H.; Su, L.; Xiao, H.; Nahrstedt, K. Incentive Mechanism for Privacy-Aware Data Aggregation in Mobile Crowd Sensing Systems. *IEEE/ACM Trans. Netw.* **2018**, *26*, 2019–2032. [CrossRef]
20. Wang, X.; Liu, Z.; Tian, X.; Gan, X.; Guan, Y.; Wang, X. Incentivizing Crowdsensing With Location-Privacy Preserving. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 6940–6952. [CrossRef]
21. Lin, J.; Yang, D.; Li, M.; Xu, J.; Xue, G. Frameworks for Privacy-Preserving Mobile Crowdsensing Incentive Mechanisms. *IEEE Trans. Mob. Comput.* **2017**, *17*, 1851–1864. [CrossRef]
22. Wang, J.; Tang, J.; Xue, G.; Yang, D. Towards Energy-Efficient Task Scheduling on Smartphones in Mobile Crowd Sensing Systems. *Comput. Netw.* **2017**, *115*, 100–109. [CrossRef]
23. Zhao, Y.; Li, Y.; Wang, Y.; Su, H.; Zheng, K. Destination-Aware Task Assignment in Spatial Crowdsourcing. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM'17, Singapore, 6–10 November 2017; pp. 297–306.
24. To, H.; Shahabi, C.; Kazemi, L. A Server-Assigned Spatial Crowdsourcing Framework. *ACM Trans. Spat. Algorithms Syst.* **2016**, *1*, 2. [CrossRef]
25. Wu, P.; Ngai, E.W.T.; Wu, Y. Toward a Real-Time and Budget-Aware Task Package Allocation in Spatial Crowdsourcing. *Decis. Support Syst.* **2018**, *110*, 107–117. [CrossRef]

26. ul Hassan, U.; Curry, E. Efficient Task Assignment for Spatial Crowdsourcing: A Combinatorial Fractional Optimization Approach with Semi-Bandit Learning. *Expert Syst. Appl.* **2016**, *58*, 36–56. [CrossRef]
27. Chen, J.; Yang, J. Maximizing Coverage Quality with Budget Constrained in Mobile Crowd-Sensing Network for Environmental Monitoring Applications. *Sensors* **2019**, *19*, 2399. [CrossRef]
28. Wang, E.; Yang, Y.; Lou, K. User Selection Utilizing Data Properties in Mobile Crowdsensing. *Inf. Sci.* **2019**, *490*, 210–226. [CrossRef]
29. He, Z.; Cao, J.; Liu, X. High Quality Participant Recruitment in Vehicle-Based Crowdsourcing Using Predictable Mobility. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 2542–2550.
30. Greene, J.; Baron, J. Intuitions about declining marginal utility. *J. Behav. Decis. Mak.* **2001**, *255*, 243–256. [CrossRef]
31. Marginalism Principle. Available online: http://www.opentextbooks.org.hk/system/files/export/15/15497/pdf/Principles_of_Managerial_Economics_15497.pdf (accessed on 13 May 2019).
32. Parent, M.; Plangger, K.; Bal, A. The New WTP: Willingness to Participate. *Bus. Horiz.* **2011**, *54*, 219–229. [CrossRef]
33. Zhang, S.; Wang, S.; Xia, H.; Cheng, X. An Attack-Resistant Reputation Management System For Mobile Ad Hoc Networks. *Procedia Comput. Sci.* **2019**, *147*, 473–479. [CrossRef]
34. Li, B.; Li, R.H.; King, I.; Lyu, M.R.; Yu, J.X. A Topic-Biased User Reputation Model in Rating Systems. *Knowl. Inf. Syst.* **2015**, *44*, 581–607. [CrossRef]
35. Josang, A.; Ismail, R. The Beta Reputation System. In Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia, 17–19 June 2002; pp. 2502–2511.
36. Euclidean Distance. Available online: https://www.pbarrett.net/techpapers/euclid.pdf (accessed on 5 May 2019).
37. Dong, X.L.; Saha, B.; Srivastava, D. Less is More: Selecting Sources Wisely for Integration. *Proc. VLDB Endow.* **2012**, *6*, 37–48. [CrossRef]
38. Yang, J.; Xing, C. Data Source Selection Based on an Improved Greedy Genetic Algorithm. *Symmetry* **2019**, *11*, 273. [CrossRef]
39. Yang, Z.; Wang, G.; Chu, F. An Effective GRASP and Tabu Search for the 0-1 Quadratic Knapsack Problem. *Comput. Oper. Res.* **2013**, *40*, 1176–1185. [CrossRef]
40. Lechowicz, P.; Walkowiak, K.; Klinkowski, M. Greedy Randomized Adaptive Search Procedure for Joint Optimization of Unicast and Anycast Traffic in Spectrally-Spatially Flexible Optical Networks. *Comput. Netw.* **2018**, *146*, 167–182. [CrossRef]
41. Rutenbar, R.A. Simulated Annealing Algorithms: An Overview. *IEEE Circuits Devices Mag.* **1989**, *5*, 19–26. [CrossRef]
42. The Real-World Data. Available online: http://www.mcm.edu.cn/html_cn/node/460baf68ab0ed0e1e557a0c79b1c4648.html (accessed on 5 May 2019).
43. Hu, T.; Xiao, M.; Hu, C.; Gao, G.; Wang, B. A QoS-Sensitive Task Assignment Algorithm for Mobile Crowdsensing. *Pervasive Mob. Comput.* **2017**, *41*, 333–342. [CrossRef]
44. Cheng, P.; Lian, X.; Chen, Z.; Fu, R.; Chen, L.; Han, J.; Zhao, J. Reliable Diversity-Based Spatial Crowdsourcing by Moving Workers. *Proc. VLDB Endow.* **2015**, *8*, 1022–1033. [CrossRef]
45. Nguyen, P.H.; Wang, D.; Truong, T.K. A New Hybrid Particle Swarm Optimization and Greedy for 0–1 Knapsack Problem. *Indones. J. Electr. Eng. Comput. Sci.* **2016**, *1*, 411–418. [CrossRef]
46. Yang, J.; Zhao, C.; Xing, C. Big Data Market Optimization Pricing Model Based on Data Quality. *Complexity* **2019**, *2019*. [CrossRef]
47. Yang, J.; Xing, C. Personal Data Market Optimization Pricing Model Based on Privacy Level. *Information* **2019**, *10*, 123. [CrossRef]