Peer-to-Peer Networking and Applications (2023) 16:1069–1087
https://doi.org/10.1007/s12083-023-01454-4

# Time window-based online task assignment in mobile crowdsensing: Problems and algorithms

Shuo Peng[1] · Kun Liu[1] · Shiji Wang[2] · Yangxia Xiang[3] · Baoxian Zhang[1] · Cheng Li[4]

Received: 4 November 2022 / Accepted: 21 January 2023 / Published online: 2 March 2023

**Abstract**
Mobile crowdsensing (MCS) has been an effective sensing paradigm by exploiting the pervasive sensor-rich mobile devices for sensor data collection. Online task assignment is an important issue for mobile crowdsensing since tasks typically arrive dynamically and need to be handled in an online manner. In this paper, we study online task assignment for maximizing the total profit of the MCS platform while satisfying the time window requirement of each task. We first describe the crowdsensing model and then study the online task assignment in the following two different scenarios: (1) user-offline-arriving scenario, where all users are fully available throughout the whole sensing period and their movements are fully planned by the platform; (2) user-online-arriving scenario, where users arrive and depart dynamically and each user has a specific participatory time window for task executions. For the former scenario, we propose a benchmark algorithm and also an online heuristic algorithm. The benchmark algorithm tries to provide a best-case performance by assuming all future task arrival information is known in advance. The online algorithm adopts bipartite-matching-based strategy for task assignment and further performs minimal detour based data offloading for reducing the data upload cost, whenever possible. For the latter scenario, we propose an effective online algorithm, which adopts a maximum-profit-first strategy for task assignment and also minimal detour based data offloading for reduction of data upload cost whenever applicable. For all the proposed algorithms, we present their detailed design and deduce their time complexities. Extensive simulations are conducted and the results demonstrate that our proposed algorithms can largely increase the total profit of the platform as compared with existing work.

**Keywords** Mobile crowdsensing · Online task assignment · Data offloading

✉ Baoxian Zhang
bxzhang@ucas.ac.cn

Shuo Peng
pengshuo17@mails.ucas.ac.cn

Kun Liu
liukun181@mails.ucas.edu.cn

Shiji Wang
wsj978418128@163.com

Yangxia Xiang
18701539583@163.com

Cheng Li
licheng@mun.ca

1 Research Center of Ubiquitous Sensor Networks, University of Chinese Academy of Sciences, Beijing 100049, China

2 Beijing Aerospace Measurement and Control Technology Co., Ltd., Beijing 100041, China

3 Information and Communication Department, Army Academy of Armored Forces, Beijing 100072, China

4 Faculty of Engineering and Applied Science, Memorial University, St. John's, NL A1B 3X5, Canada

## 1 Introduction

With the pervasiveness of smart devices and rapid development of new wireless communication techniques, Mobile crowdsensing (MCS) [1, 2] has been an effective sensing paradigm. In mobile crowdsensing, users can utilize their carried mobile devices to complete various sensing tasks. Compared with other Internet of Things (IoT) paradigms, mobile crowdsensing has good scalability, large spatial–temporal coverage, and high sensing quality. Nowadays, mobile crowdsensing has been widely used in many scenarios such as monitoring urban traffic, detecting surrounding air quality, identifying noise pollution level, and indoor localization, etc. (see [3–8]).

Task assignment is a critical issue in mobile crowdsensing. Recently, much work has been done for improving the task assignment performance [9–18]. The composition of a typical crowdsensing system is generally as follows: mobile users, task requesters, and service platform. The service platform is responsible for organizing and managing the crowdsensing system. It receives tasks from task requesters and assigns the tasks to suitable mobile users. According to the sensing mode of users, mobile crowdsensing can be divided into the following two types [9]: 1) opportunistic sensing; 2) participatory sensing. For opportunistic sensing, users follow their daily routes and perform tasks in an opportunistic manner and their routes are in general determined by the users' habits and customs and will not change due to the locations of tasks. For participatory sensing, users are required to go directly to the locations of their undertaken tasks for task executions. In this case, their routes are fully or partially planned by the service platform. According to the task arrival pattern, task assignment can be divided into offline task assignment and online task assignment, where the former assumes all the tasks arrive before the task assignment while the latter assumes tasks arrive at the platform dynamically and are handled in an online manner. In this paper, we focus on studying online task assignment in participatory sensing. The main challenge in this case is how to design efficient task assignment algorithms for maximizing the platform profit while considering the time sensitivity of tasks, distribution of users, and also uncertainty of the task arrivals and their distributions.

In the literature, some online task assignment algorithms (e.g., [19–22]) have been proposed to enable efficient participatory sensing. These studies all assume that tasks have specific time window(s), and only users who can reach the location of a task within the task time window are eligible to complete the task. However, these algorithms did not consider the required task performing time in their designs. They mainly focused on simple sensing activities (such as photo taking) and assumed that each of the tasks can be completed immediately once a user reach the corresponding task location. In practice, the task performing time of many tasks (e.g., traffic flow monitoring, noise detection) are often non-negligible. These tasks last for a certain period of time and require continuous sensing data. Although some work (e.g., [23–25]]) took the task performing time into account, they focused on opportunistic crowdsensing and further did not consider how to reduce the task result upload cost (e.g., when cellular traffic is used) in their algorithm design. Thus, effective data offloading via complimentary WiFi based Internet accessing is very important to improve the profit of the service platform, which is also a research focus of our work in this paper.

In this paper, we study time window-based online task assignment in participatory sensing. The objective is to maximize the service platform's profit. We divide the sensing period into multiple equal-length timeslots. At the beginning of each timeslot, task requesters submit task request(s) to the service platform. Each of the tasks has a specific location for data collection and further is associated with an executing time window, which represents the duration of the task. More specifically, a recruited user must continuously conduct data collection within the specified time window of target task at the corresponding task location. According to different ways of user arrivals, in this paper, we consider two user arrival scenarios/patterns: user-offline-arriving scenario and user-online-arriving scenario. The former means that all users arrive at the platform before the task assignment and are fully available all the time. The latter means that users arrive and leave dynamically and each of them is available for task execution just for a certain period of time. We introduce the crowdsensing system under study and describe the above two scenarios in details according to the user arrival patterns. We formulate the optimization problem for either scenario and accordingly propose efficient algorithms for both scenarios. Our main contributions in this paper are listed as follows:

- For the user-offline-arriving scenario, we first formulate the corresponding profit-maximization problem and then propose a benchmark algorithm and an online task assignment algorithm. The benchmark algorithm tries to provide a best-case performance by assuming all future task arrival information is known in advance. The online algorithm performs per-slot based bipartite-matching by using the Kuhn-Munkres algorithm for task assignment and adopts minimal detour based data offloading for maximally reducing the data upload cost, if such offloading is beneficial. We present detailed designs for both algorithms and deduce their computational complexities.
- For the user-online-arriving scenario, we formulate the profit-maximization problem for this case and propose an online task assignment algorithm, which adopts a greedy maximum-profit-first strategy for task assignment and performs minimal detour based data offloading for effectively reducing data upload cost, if possible. We present its detailed algorithm design and deduce its computational complexity.
- We conduct extensive simulations for performance evaluation and the results validate the high efficiency of our proposed algorithms.

In our earlier work [10], we focused on studying the user-offline-arriving scenario and proposed effective online task assignment algorithms. Compared with [10], the following new contributions are made in this paper: 1) For the user-offline-arriving scenario, we present a benchmark algorithm, which is to provide a best-case performance for comparison purpose; 2) For the user-online-arriving scenario, which was not investigated in [10], we here formulate the optimization problem under study for this case and accordingly propose an effective online task assignment algorithm.

The rest of this paper is organized as follows. In Section 2, we give a brief review of related work. In Section 3, we introduce the system model. In Sections 4 and 5, we formulate the problems for different scenarios and accordingly propose effective polynomial algorithms. In Section 6, we perform extensive simulations for performance evaluation. Finally, in Section 7, we conclude this paper.

## 2 Related work

Much work has been done for achieving high task assignment performance in mobile crowdsensing. According to whether the tasks are time sensitive, existing task assignment algorithms can be divided into two categories: time insensitive task assignment and time sensitive task assignment. In the following, we will introduce typical work falling into either category.

### 2.1 Time insensitive task assignment

For time insensitive task assignment, tasks do not have specific start time and end time, and can be executed at any time. According to the sensing mode used, existing work in this area can be further categorized into participatory time insensitive task assignment algorithms and opportunistic time insensitive task assignment algorithms.

Some typical work for participatory time insensitive task assignment is as follows. In [11], the authors studied how to maximize the total profit subject to budget constraint. They first proved the NP-hardness of this problem and then proposed an approximation algorithm, which decomposes the original problem into several sub-problems. In [26], the authors tried to minimize the incentive payout while ensuring the task quality. They accordingly designed an approximate algorithm based on greedy strategy for task assignment. Ref. [27] studied a platform-centric task assignment problem and proposed a genetic algorithm. Ref. [28] studied participatory sensing in two different scenarios according to the difference in number of tasks and number of users. The first scenario is when number of tasks are more than that of users. For this scenario, the paper studied how to minimize the total traveling distance of users and also maximize the total number of completed tasks. Then two optimal algorithms are designed for this scenario. The second scenario is that there are more users than tasks. The design goal for this scenario is to minimize the payment to users and also minimize the recruited users' traveling distance. For this scenario, two heuristic algorithms were proposed.

Some typical work for opportunistic time insensitive task assignment is as follows. In [29], Peng et al. studied AP-assisted online task assignment while minimizing the average finishing time and largest finishing time of all tasks, respectively. They first derived the expected finishing time of a task by considering the opportunistic exchanging of tasks/results via APs, then they proposed two online algorithms which utilize the communication ability of APs to minimize the average makespan and largest makespan of tasks, respectively. Ref [30]. was aimed at maximizing task quality based utility subject to incentive constraint. The authors first proved the NP-hardness of the utility maximization problem and then proposed an approximate algorithm by leveraging the submodular property of the objective function they adopted.

### 2.2 Time sensitive task assignment

Much work for time sensitive task assignment assumes that each task is associated with a specific time window and needs to be executed during its specified time window. Among the existing work in this aspect, some did not consider the task execution time and some other did. For the former, users passing by the sensing location of a task within the corresponding time window can perform the task in negligible time. For the latter, users need to execute the task continuously within the corresponding time window at given location.

Some existing work without considering task execution time is as follows. In [20], the authors studied the maximization of total task qualities subject to individual user travel distance constraints. They proposed four online heuristic algorithms for this case. In [22], the authors focused on maximizing the platform's profit. For this problem, they proposed one offline algorithm and three online algorithms.

Existing work considering the task execution time is as follows. Ref [23]. was aimed to minimizing the social cost of task performing in opportunistic sensing while ensuring the completion of tasks. The authors formulated this problem and considered two different scenarios based on whether users have multiple time windows for task performing. They proposed corresponding user selection algorithms for different scenarios. In [24], Xu et al. aimed to minimizing the payment for recruiting users. They modeled the recruitment scenario as a reverse auction process. Then an incentive mechanism was proposed for rewarding selected users. In [25], Sun et al. studied the maximization of data benefit while ensuring that the task period can be covered by recruited users. Then they designed an optimal user selection algorithm for this problem. One big issue in the above work is that they did not consider how to minimize the upload cost of task result. When the data amount of task result is large, such lack of consideration on upload cost can largely affect the profit of the platform. In this paper, we shall consider use of data offloading, whenever possible, to reduce the upload cost as much as possible.

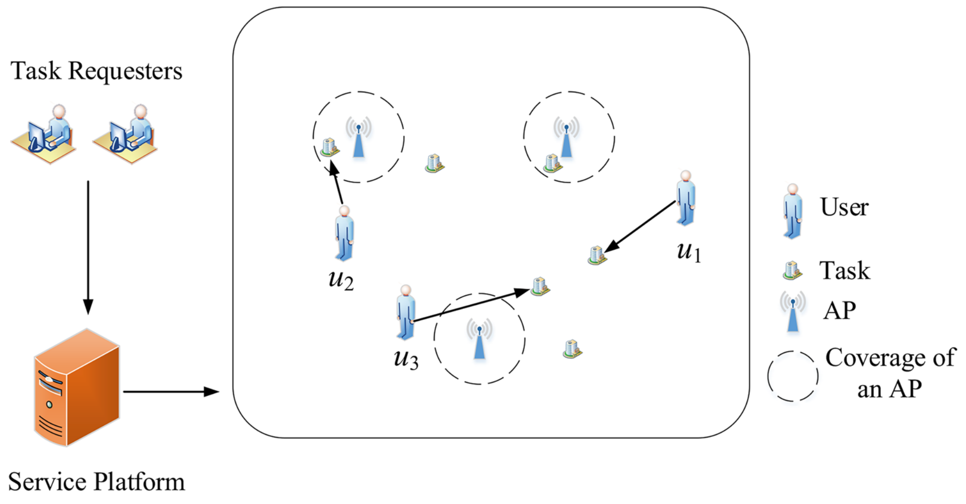


**Fig. 1** Overview of the mobile crowdsensing system

## 3 System model

This section first introduces the crowdsensing system under study, and then describes various models in the crowdsensing system in details.

### 3.1 System overview

Figure 1 gives an overview of the mobile crowdsensing system under study, which is composed of the following major components: users, APs, task requesters, and service platform. The MCS system works in an online fashion. Time is divided into $q$ equal-length timeslots, denoted as $L=\{l_1, l_2, \ldots, l_q\}$. Tasks arrive at the platform at the beginning of timeslots. The tasks are time window-specified, location-dependent, and are published by the requesters on the platform. To ensure the timely completion of the tasks, the platform needs to recruit certain mobile users. Since each task has strong data integrity requirement, a recruited user needs to continuously execute the task at designated task location. In order to have more qualified user candidates for task executions, each task requester should submit task requests to the platform $T$ timeslots ($T\geq 1$) before their desired data collections start. For example, if a task actually starts from timeslot $l$, then the corresponding task request should be submitted to the platform at timeslot $l-T$. Here, $T$ is in general a small integer. Upon receipt of such a task, the platform is supposed to immediately assign the task to an available user, if possible. A user with assigned task(s) needs to sequentially visit the task location(s) during their respective time period(s) to accomplish the corresponding task(s), respectively. After a task is accomplished, the task result needs to be uploaded to the platform in a certain period of time. Here, we assume that a user must submit the result of a task within $K$ timeslots once completing a task. After submitting the task results, a user can get certain incentive from the platform. Finally, the requester can get the task result from the platform and should give corresponding reward to the platform. The amount of reward provided for the execution of a task is determined by the task requester according to the value of the task and is known at the platform. In this way, the platform is able to earn profit by recruiting users for task completions. It needs to be pointed out that if the execution of a task will cause negative profit at the platform, the platform will reject such task without recruiting any user for its completion. This design choice is reasonable because a platform is supposed to be unwilling to take unprofitable tasks. Table 1 lists major notations used hereafter.

According to the ways users arrive at the platform, in this paper, task assignment is studied in the following two scenarios.

*User-Offline-Arriving scenario*: All users are assumed to have arrived at the platform before the task assignment and each of them will be available all the time for task executions. In the whole sensing period, the movements of all the users are fully planned by the service platform for task accomplishment.

*User-Online-Arriving scenario*: Users arrive at the platform dynamically. Specifically, each user is associated with the following information: a start location, a predetermined destination, a constant moving speed, and the deadline for the user to reach his destination. A user can take detour to accomplish certain task(s) provided that he can still reach his destination before the given deadline.

### 3.2 System models

In this subsection, we introduce the following four models in the crowdsensing system: user model, task model, data upload model, and incentive model.

**Table 1** Notations used

| Notations | Definitions |
|---|---|
| $u_i$, $U$ | A user $u_i$ and the set of all users |
| $j_l^k$, $J_l$, $J$ | The *k*th task arrived in timeslot *l*, the set of tasks arriving in timeslot *l*, and the set of all tasks |
| $loc_l^k$, $s_l^k$, $e_l^k$ | Location, start timeslot and end timeslot of task $j_l^k$ |
| $a_i$, $f_i$ | Arriving timeslot and leaving timeslot of user $u_i$ |
| $v_i$ | Moving speed of user $u_i$ |
| $r_l^k$ | Reward of task $j_l^k$ |
| $b_l^k$ | Bonus for a user to finish task $j_l^k$ |
| $t_l^k$ | Time window length of task $j_l^k$ |
| $m$ | Total number of users |
| $n$ | Average number of task arrivals in a timeslot |
| $w$ | Total number of APs |
| $n_l$ | Number of task arrivals in timeslot *l* |
| $q$ | Total number of timeslots |
| $K_d$ | Unit distance cost for traveling |
| $D_t$ | Data amount of unit time |
| $C$ | Unit upload cost, i.e., cost for use of unit cellular traffic |
| $K_t$ | Unit task bonus, i.e., bonus for execution of unit task |
| $K$ | Maximum upload deadline for task result uploading |
| $x_{li}^k$ | Binary decision variable between user $u_i$ and task $j_l^k$ |
| $x_l$ | Task assignment result of timeslot *l* |

**User model** Denote the set of users in the system by $U = \{u_1, u_2, \ldots, u_m\}$. All the users have registered at the platform. We assume each user carries a smart device and all the devices are equal and eligible for accomplishing any task published by the platform. The location of a user can be obtained via equipping a GPS receiver at the user side and such information needs to be periodically reported to the platform once a user logins to the system. Denote the moving speed of user $u_i \in U$ as $v_i$. It is assumed that any user can execute at most one task at a time. If a user is chosen to undertake a task, he must move to the corresponding task location for the task execution before the task starts. For the user-online-arriving scenario, the following additional information is needed: Each user needs to indicate his target destination and also a deadline for him to reach the destination. Each user must report all the travel related information to the platform upon his arrival.

**Task model** Denote the set of all tasks as $J$. Since there are totally $q$ timeslots, we have $J = \{J_1, J_2, \ldots, J_q\}$, where $J_l \in J$ $(1 \le l \le q)$ represents the set of tasks arrived in timeslot $l$. Let $n_l$ represent the number of task arrivals in timeslot $l$, we have $J_l = \{j_l^1, j_l^2, \ldots, j_l^{n_l}\}$. A task $j_l^k \in J_l$ can be represented by a tuple $\{loc_l^k, s_l^k, e_l^k, r_l^k\}$, where $loc_l^k$ is the location of the task, $s_l^k$ is the start timeslot of the task, $e_l^k$ is the end timeslot of the task, $r_l^k$ is the reward of the task that the platform can receive from the task requester.The task execution time window associated with $j_l^k$ is $[s_l^k, e_l^k]$. Obviously, we have $s_l^k = l + T$ and $e_l^k \ge s_l^k$. To ensure the data integrity of a task, a user who performs the task must collect sensing data continuously in the required time window at the corresponding task location. We assume the time window length is always an integer number of timeslots. Therefore, the execution of a task $j_l^k$ should start at the beginning of $s_l^k$ and terminate at the end of $e_l^k$. We assume each task only needs to be executed once and it can be accomplished by any user. Once the task is completed, the task result must be uploaded to the platform within $K$ timeslots.

**Data upload model** After a task is completed, the user should upload the sensing result of the task. In the system, two ways for data uploading can be used: 1) cellular networks, 2) WiFi based access points (AP) (with preference if applicable). The former will cause certain fee while the latter is complimentary. The whole sensing area is assumed to be fully covered by cellular services. Thus a user can always upload his collected sensing data via cellular traffic at any time, but with a certain uploading cost. For WiFi based APs, they are randomly distributed in the map and their service coverage is limited. A user can only upload task results through an AP when he is in or passes by the coverage of the AP. Due to the high-speed properties of 4G/5G/WiFi, similar to [31], we in this paper assume the data rate for result uploading is very high and thus the corresponding data uploading time is negligible compared with the travel time of users among different task locations. The choice of data uploading mode depends on the following aspects: the deadline for data uploading, the location of the completed task, and the coverages of the deployed APs.

**Incentive model** In general, the total incentive that a user can obtain for performing a task is affected by the cost caused in the user's execution of the task. In order to effectively attract users to perform tasks, we assume that each user can get certain bonus once he completes a task. Therefore, the incentive defined here includes travel cost, upload cost, and bonus. Accordingly, the incentive provided to user $u_i$ for his accomplishment of a task $j_l^k$ is calculated as below:

$$Incentive_{li}^k = cost\_dist_{li}^k + upload_l^k + b_l^k, \tag{1}$$

where $cost\_dist_{li}^k$ is the travel cost for task accomplishment, $upload_l^k$ is the uploading cost for the result of task $j_l^k$, and $b_l^k$ is the bonus that the platform gives a user for his accomplishment of the task. We in this paper adopt linear functions for the computation of incentive. In fact, how the incentive for task execution is computed can be adjusted according to specific application scenarios, which does not affect the usability of our task assignment algorithm. More details about the incentive computation are as follows.

The travel cost is assumed to be proportional to the travel distance, which can be computed as follows:

$$cost\_dist_{li}^{k} = d_{li}^{k} * K_d, \tag{2}$$

where $d_{li}^{k}$ is the travel distance for user $u_i$ to reach task $j_l^k$, and $K_d$ is the travel cost per unit distance.

Before defining the upload cost and bonus, we first define the time window length of a task. For a task $j_l^k \in J_l$, its time window length is calculated as below:

$$t_l^k = e_l^k - s_l^k + 1. \tag{3}$$

In practice, it is plausible to treat the time window length of a task as its task workload.

The upload cost of task result is thus proportional to the task's time window length when cellular traffic is used or zero when WiFi based Internet accessing is used. Accordingly, the upload cost $upload_l^k$ for task $j_l^k$ is computed as below:

$$upload_l^k = \begin{cases} t_l^k * D_t * C & Cellular\ Network \\ 0 & AP \end{cases}, \tag{4}$$

where $D_t$ is the amount of data collected in unit time, and $C$ is the price for uploading unit data amount by using cellular traffic.

The bonus that a user can obtain for performing a task is also proportional to time window length of the task, which is calculated as below:

$$b_l^k = t_l^k * K_t, \tag{5}$$

where $K_t$ represents the bonus for performing a task with unit time duration.

The profit that the platform can earn for the execution of task $j_l^k$ thus equals the corresponding reward $r_l^k$ minus the incentive provided to its executing user $u_i$, which is as follows:

$$profit_{li}^{k} = r_l^k - Incentive_{li}^{k}. \tag{6}$$

## 4 User-offline-arriving task assignment: problem and algorithms

In this section, we study the task assignments in the user-offline-arriving scenario such that all the users are fully available throughout the whole sensing period and all their trajectories are purely planned for task accomplishments as necessary.

We first formulate the profit maximization problem for this scenario and demonstrate the NP-hardness of this problem. To address this issue, we design a benchmark algorithm and an online algorithm. The benchmark algorithm assumes that all future task arrival information is known in advance and thus provides a theoretical best case performance. The online algorithm assumes dynamic task arrivals. It adopts per-slot-based bipartite graph matching algorithm for task assignment and reduces the upload cost via data offloading (referred to as BMA-RUA). A common feature of these two algorithms is that both of them iteratively optimize the task assignment in each individual timeslot, starting from the first timeslot, by using bipartite graph matching. The key difference between them is as follows. The benchmark algorithm adopts earliest end timeslot first strategy in the task assignment due to its assumption of full availability of all future task information. In contrast, the online algorithm works in a task-arrival-triggered manner and adopts earliest start timeslot first strategy in the task assignment due to its assumption of dynamic task arrivals.

### 4.1 Problem formulation

Denote the task assignment result between user $u_i$ and task $j_l^k$ as $x_{li}^k$. If task $j_l^k$ is assigned to user $u_i$, we have $x_{li}^k = 1$, otherwise, $x_{li}^k = 0$. Then the profit maximization problem for the user-offline-arriving scenario is formulated as follows:

$$max \sum_{l=1}^{q} \sum_{i=1}^{m} \sum_{k=1}^{n_l} profit_{li}^{k} * x_{li}^{k}, \tag{7}$$

$$\text{s.t. } \textstyle\sum_{i=1}^{m} x_{li}^{k} \leq 1, \forall l \in L, j_l^k \in J_l. \tag{8}$$

$$\sum_{k=1}^{n_l} x_{li}^{k} \leq 1, \forall l \in L, u_i \in U. \tag{9}$$

$$\frac{d_{li}^{k}}{v_i} * x_{li}^{k} \leq T, \forall u_i \in U, l \in L, j_l^k \in J_l \tag{10}$$

$$x_{li}^{k} \in \{0,1\}, \forall u_i \in U, l \in L, j_l^k \in J_l. \tag{11}$$

Constraint (8) ensures that a task can be assigned to at most one user. Constraint (9) ensures that a user can perform at most one task in any timeslot. Constraint (10) guarantees a user can only complete those tasks, whose sensing locations can be reached by the user within $T$ timeslots. In (7)-(11), since $x_{li}^k$ is a binary decision variable, the task assignment problem in user-offline-arriving scenario is a 0–1 integer linear programming problem, which is known to be NP-hard.

Obviously, seeking the global optimal solution using exhaustive search is very time consuming. Next, we study how to achieve optimal task assignment for each individual timeslot and accordingly design two heuristic algorithms.

**Algorithm 1** Benchmark Algorithm.

```
Require:
    $U$, $J$, $L$;
Ensure:
    $profit$, $x_l = \{x_{l1}^1, ..., x_{lm}^{n_l}\}$, $\forall l \in L$;
 1: Initialize $profit \leftarrow 0$; $x_l = \{0, ..., 0\}$, $\forall l \in L$.
 2: Sort all tasks by end timeslot in the increasing order.
 3: for $l$ from 1 to $q$ do
 4:     Get the set of tasks $J_l^*$ ending in timeslot $l$.
 5:     Construct a bipartite graph $G$, where the vertices
        are $U$ and $J_l^*$, and the edge weights are the profits
        for the users to complete the tasks.
 6:     Match $U$ and $J_l^*$ with Kuhn-Munkres algorithm for
        maximum profit in $G$.
 7:     for each user $u_i \in U$ do
 8:         if there exists any task $j_{l^*i}^k$ assigned to user $u_i$
            then
 9:             $x_{l^*i}^k \leftarrow 1$
10:             $profit \leftarrow profit + profit_{l^*i}^k$
11: return $profit$, $x_l = \{x_{l1}^1, ..., x_{lm}^{n_l}\}$, $\forall l \in L$.
```

**Algorithm 1.** Benchmark Algorithm

## 4.2 Benchmark algorithm

In this subsection, we present a benchmark algorithm, which assumes that all the future task information is known in advance. That is, all the user- and task-related information are known in advance. Use of brute-force or branch-and-bound search for the optimal solution could be extremely time-consuming. Instead of pursuing such global optimal solution, we here seek for optimal task assignment for each individual timeslot.

The main idea behind the benchmark algorithm is as follows. It first ranks all the tasks in an increasing order of end timeslots. For the task assignment, it adopts an *earliest end timeslot first* strategy, i.e., the tasks with earliest end timeslot are assigned first. Such a greedy strategy can leave more time for users to perform tasks in the future. For each timeslot, it makes optimal matching between the users and the tasks ending in the timeslot. To achieve the per-slot optimal matching, the task assignment problem is transformed into a bipartite graph matching problem by constructing a bipartite graph. For a timeslot, tasks ending in the timeslot and all the users can be regarded as the vertices of the bipartite graph. For each user-task pair, it is connected by an edge, whose weight is the profit that the platform can earn from the corresponding user-task pair, which can be calculated by using (6). Specifically, for a task $x$, if a user $u_i$ cannot reach a task $x$'s location in time (i.e., reaching there before the start timeslot of $x$, by starting the movement immediately if the user is not currently undertaking any task or after the user finishes his last undertaken task), or his accomplishment of the task does not bring positive profit for the platform, the weight of the corresponding edge between them will be set to 0.[1] Then, the Kuhn-Munkres algorithm [32] is used to achieve per-slot optimal bipartite graph matching. Obviously, such per-slot optimization cannot yield the global optimal solution of the whole sensing period.

Algorithm 1 gives the detailed procedure of the benchmark algorithm. The variable *profit* records the total profit that the platform earns. Line 1 initializes necessary variables. Line 2 sorts all tasks according to their end timeslots. Lines 3–10 perform task assignment. Specifically, lines 5 and 6 construct the bipartite graph and perform the corresponding matching. Lines 7–10 update all variables. Note that the variable $l$ in line 1 means task ending timeslot while the variable $l^*$ in lines 8–10 mean the arrival timeslot of a task ending at timeslot $l$.

The time complexity of the benchmark algorithm is deduced as follows. Assuming that the average number of tasks ends at a timeslot is $n$, the average number of total task arrivals will be $qn$, where $q$ is the total number of timeslots. So line 2 takes $O(qn)$ time when counting sorting is used.[2] Since the bipartite graph matching in line 6 takes $O((m+n)^3)$ time, lines 4–10 take $O((m+n)^3)$ time in total. Thus lines 3–10 take $O(q(m+n)^3)$ time. Therefore, we have the benchmark algorithm has a total time complexity of $O(q(m+n)^3)$.

## 4.3 BMA-RUA algorithm

The BMA-RUA algorithm is an online task assignment algorithm, where tasks arrive dynamically. Different from the benchmark algorithm, it adopts earliest start timeslot first strategy. For each timeslot, BMA-RUA includes two main steps: task assignment and data offloading.

**Step 1: Task assignment**. It works in a task-arrival-triggered manner and adopts earliest start timeslot first strategy in the task assignment due to the dynamic task arrival property. It makes task assignment decisions for all tasks arrived at the beginning of each timeslot. At the beginning of a timeslot, since the information of all users and the tasks already arrived at this time are both known, the task assignment problem can be addressed by using bipartite graph matching similar to Algorithm 1. According to the matching result, the platform can get the maximum profit of each individual timeslot. Accordingly, per-slot based optimal assignment is achieved in such an online manner.

[1] Note that the weights of edges in a bipartite graph are all calculated this way in later algorithms proposed in this paper, whenever applicable.

[2] In this paper, we focus on scenarios where the duration of a timeslot is quite long such that the mean number of task arrivals in a timeslot is much larger than one. In this case, the total number of timeslots will be much smaller than the (mean) total number of tasks (i.e., $q << qn$). Thus, the applicability condition for using counting sorting holds.

**Step 2: Data offloading**. At the end of a timeslot, upon completion of a task (if any), the task result should be submitted to the platform within a certain period of time. Due to the limited coverage of the APs, submitting the task results directly to the platform via the cellular network will bring more upload cost. In order to increase the profit of the service platform, full use of the AP coverages is encouraged to reduce the upload cost. For this purpose, we largely utilize the service availability of the APs for data offloading.

In BMA-RUA, minimal detour based task results offloading is adopted to maximally reduce the upload cost. Here, minimal detour means with minimal extra travel distance is pursued for WiFi based data offloading if such offloading is beneficial. When multiple such choices exist, we choose the one leading to the minimal extra travel distance. Accordingly, upon the accomplishment of a task, there are the following two cases for a user to perform data offloading.

1) If the user has been assigned a new task

In this case, the user can choose to submit most recently collected task results through an AP on his way to the new task location, if possible. To find such an AP, we need to seek a shortest route connecting the user, a point in the AP's service range, and the target task. For visiting the AP, we actually need to seek a point on the perimeter of the AP's circular service range, which results in the shortest travel distance while meeting the following eligibility conditions: 1) the user can reach the chosen point in $K$ timeslots, and further 2) he can reach the next task location before the task starts. The point leading to the shortest distance can be calculated by using geometric methods. Specifically, suppose the user is currently at location $A$ and plans to go to next task location $B$, while trying to detour to a point on the service range of an AP located at $O$, the problem is to find such a node $X$, while leading to the shortest distance $A$–$X$-$B$. There are two cases:

Case 1: If the segment $AB$ penetrates the AP's service range, then the first intersection point is $X$ (see Fig. 2a); Case 2: We need to find a point $X$ such that the half-line $OX$ equally divides $\angle AXB$ (see Fig. 2b). Finding such a location $X$ in this case can be transformed to the solving a quartic equation with one unknown quantity.[3] For more details regarding how to determine the coordinate of $X$, please see the Appendix.

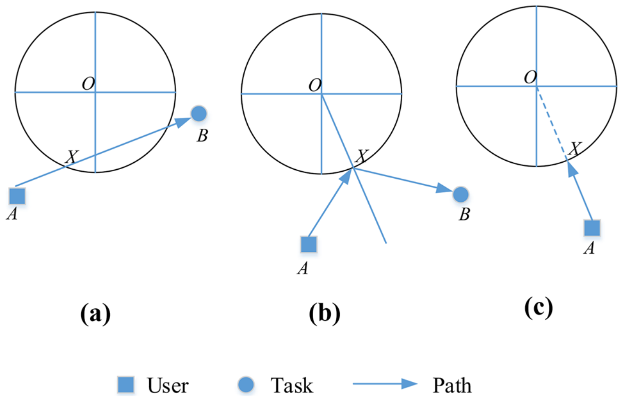


Fig. 2 Illustration of various cases for reducing upload cost via data offloading. Point $O$ is where an AP is located

When multiple such APs exist, we need to find the AP and corresponding point, which lead to the shortest distance among all the choices while meeting the above eligibility conditions. In this way, on the way moving to the next task location, the task result can be uploaded via the chosen AP.

2) If the user is not currently assigned any new task

In this case, he can move to a closest AP's service range for data offloading if he can reach there within $K$ timeslots (see Fig. 2c). Upon reaching there, he can upload his held task results through the AP.

Note that one additional precondition for the above data offloading to be applicable is that it is cost effective, i.e., it can bring positive profit to the platform.

Algorithm 2 shows the procedure of BMA-RUA. $task\_list_i$ represents the set of tasks which have been completed by user $u_i$ but the task results have not been submitted yet. $upload_i^*$ represents the *cellular*-based upload cost (i.e., suppose cellular traffic is used) of all those *offloadable* tasks in $task\_list_i$, which can be offloaded before their upload deadlines. $cost_{extra}$ represents the extra travel cost caused by data offloading, if any. Note that the values of $upload_i^*$ and $cost_{extra}$ can change with the routes taken for data offloading.

Algorithm 2 works as follows. Line 1 initializes necessary variables. Lines 2–21 are for per-slot based task assignment (see lines 3–5), data offloading (see lines 6–19), and uploading via cellular traffic (see lines 20–21). The first is executed at the beginning of a slot while the latter two are executed at the end of a slot. In lines 6–19, task accomplishment triggered data offloading (if applicable) is used. Specifically, if the just-accomplished task location is covered by an AP, immediate offloading is performed (see line 19), otherwise

[3] This problem is also known as the pilgrimage to castrum problem, which can be briefly described as follows. There was a vendor, who worked at a bazaar. Each day he went to the bazaar from his home. But before reaching the bazaar, he always went first to a circular castrum to worship the statue of Apollo, which could be done at any boundary point of the castrum. The problem is thus to find a worship point which minimizes the total travel distance from his home to the worship point and then all way to the bazaar.

```
Algorithm 2 BMA-RUA.
Require:
    $U$, $J$, $L$, $K$;
Ensure:
    $profit$, $x_l = \{x_{l1}^1, ..., x_{lm}^{n_l}\}$, $\forall l \in L$;
1:  Initialize $profit \leftarrow 0$; $task\_list_i \leftarrow \phi, \forall u_i \in U$; $x_l \leftarrow \{0, ..., 0\}$, $\forall l \in L$.
2:  for $l$ from 1 to $q$ do
      // Step 1: Task assignment
3:    Get the set of tasks $J_l$ arrived in timeslot $l$.
4:    Construct a bipartite graph $G$, where the vertices are $U$ and $J_l$, and the edge weights are the profits for the users to complete the tasks.
5:    Match $U$ and $J_l$ with Kuhn-Munkres algorithm for maximum profit in $G$, and accordingly update $x_l$.
      // Step 2: Data offloading
6:    for each user $u_i \in U$ who has just completed a task at the end of timeslot $l$ do
7:      if user $u_i$ is not covered by any AP then
8:        if user $u_i$ has been assigned another task $j_{l^*}^k$ to be executed in a near future timeslot then
9:          Find the shortest $feasible$ route ($u_i$–AP-$j_{l^*}^k$), and obtain corresponding $cost_{extra}$ and $upload_i^*$.
10:         if such a shortest $feasible$ route exist then
11:           Go along the above route for data offloading and update $profit$ and $task\_list_i$.
12:         else
13:           Go directly to $j_{l^*}^k$ for next task execution and update $profit$ and $task\_list_i$.
14:       else
15:         Find the shortest $feasible$ route ($u_i$–AP), and obtain corresponding $cost_{extra}$ and $upload_i^*$.
16:         if such a $feasible$ path exists then
17:           Go along the above route for data offloading and update $profit$ and $task\_list_i$.
18:     else
19:       Submit task results in hand via AP, and update $profit$ and $task\_list_i$.
      //For checking at the end of timeslot $l$
20:   for each user $u_i \in U$ do
21:     Upload task results in $task\_list_i$ that have reached the upload deadline via cellular network, and update $task\_list_i$ .
22: return $profit$, $x_l = \{x_{l1}^1, ..., x_{lm}^{n_l}\}$, $\forall l \in L$. =0
```

**Algorithm 2** BMA-RUA

we need to seek an offloading route in the form of “user-AP-NextTaskLocation” (see lines 8–13) or in the form of “user–AP” (see lines 15–17). It should be noted that only feasible offloading routes are considered, i.e., timely reaching target AP and also next task location (if applicable), and cost effectiveness (i.e., positive profit to the platform). When multiple such choices exist, the shortest one will be chosen. Lines 20–21 check whether each piece of task result in hand has reached its upload deadline. If so, the user must upload the task results via cellular network immediately.

The time complexity of BMA-RUA is as below. Lines 3–5 for bipartite graph matching takes $O((m+n)^3)$ time, where $n$ is the average number of task arrivals in a timeslot[4]. Lines 8–13 take $O(w)$ time and also lines 15–17 take $O(w)$ time. Here, we assume finding a “user-AP-NextTaskLocation” route takes constant time. Line 19 takes $O(1)$ time. Then the “for” loop between line 6 and line 19 takes $O(mw)$ time. Lines 20–21 take $O(m)$ time. Therefore, lines 3–21 take $O((m+n)^3)$ time. Since there are $q$ timeslots totally, the overall time complexity of BMA-RUA is $O(q(m+n)^3)$.

## 5 User-online-arriving task assignment: problem and algorithm

This section studies the user-online-arriving task assignment problem where users arrive and depart dynamically. We first formulate the profit maximization problem for this scenario, and then design an online heuristic algorithm. The designed algorithm is a greedy maximum profit first algorithm, which iteratively selects the user-task pair leading to the maximum profit among all user-task pairs until no such choice exists. This algorithm also adopts the data offloading strategy for reducing upload cost. We refer to this algorithm as MPF-RUA.

### 5.1 Problem formulation

In the user-online-arriving scenario, users arrive and leave dynamically. Figure 3 gives an example illustrating the dynamic user arrivals and departures in the user-online-arriving scenario.

Each user $u_i \in U$ has an arrival time, a start location, a predetermined destination, a constant moving speed, and a deadline for the user to reach the destination. On the way a user moves to his destination, he is allowed to take detour to complete one or more tasks provided that the user can reach his destination before pre-determined deadline. A user $u_i$ is associated with a time window $[a_i, f_i]$, where $a_i$

[4] It should be noted that, in the deduction of the complexity of the benchmark algorithm in the preceding subsection, we used $n$ to represent the average number of tasks *ending* in a timeslot. Here, we use $n$ to represent the average number of tasks *arriving* in a timeslot. The reason we can use $n$ to represent both variables is because, in the long term, we have the average number of tasks arriving in a timeslot equals the average number of tasks ending in a timeslot. The reason is as follows. Since the value of $T$ does not affect the conclusion, we here simply choose $T=0$. Without loss of generality, the duration of a task is assumed to be uniformly chosen from $\{1, 2, \ldots, k\}$ timeslots and the number of tasks arrived in a slot is $n$, then the number of tasks ending in a slot is due to the contribution of its preceding $k-1$ timeslots and also the current timeslot, each contributing an average number of $(1/k)n$ tasks. Obviously, the expected total number of tasks ending in a slot is also $n$.

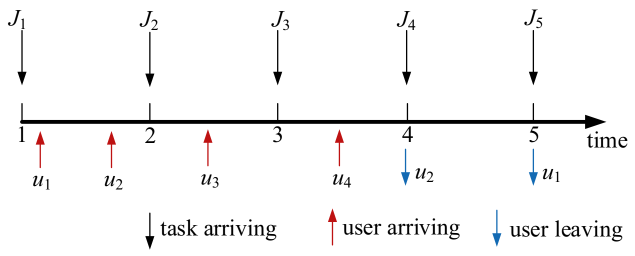


**Fig. 3** An example illustrating the user online arriving scenario

is the timeslot when he arrives at the platform, and $f_i$ is the last timeslot when he must have reached the destination. To ease the description, we assume the user must reach the destination before the end of timeslot $f_i$. The length of the time window restricts length of the detour that a user can take. Let $E_{i,last}$ represent the end timeslot of the last task that user $u_i$ undertakes. Let $T_{i,dest}$ represent the time required for user $u_i$ to travel from his last undertaken task location to his destination. When a user is not assigned any task, we treat his start location as his last task location. We use auxiliary variable $\phi_{li}$ to indicate whether user $u_i$ performs task or not in timeslot $l$. We have $\phi_{li} = 0$ if $u_i$ performs a task in timeslot $l$, otherwise $\phi_{li} = 1$. Then the task assignment problem in user-online-arriving scenario is formulated as follows.

$$max \sum_{l=1}^{q} \sum_{i=1}^{m} \sum_{k=1}^{n_l} profit_{li}^k * x_{li}^k \tag{12}$$

$$\text{s.t. } \left(a_i - s_l^k\right) * x_{li}^k \leq 0, \forall u_i \in U, l \in L, j_l^k \in J_l \tag{13}$$

$$\left(f_i - e_l^k\right) * x_{li}^k \geq 0, \forall u_i \in U, l \in L, j_l^k \in J_l \tag{14}$$

$$a_i + \sum_{l=a_i}^{E_{i,last}} \left[\phi_{li} + \sum_{k=1}^{n_l} x_{li}^k * t_l^k\right] + T_{i,dest} \leq f_i, \forall u_i \in U \tag{15}$$

$$\sum_{i=1}^{m} x_{li}^k \leq 1, \forall l \in L, j_l^k \in J_l \tag{16}$$

$$\sum_{k=1}^{n_l} x_{li}^k \leq 1, \forall l \in L, u_i \in U \tag{17}$$

$$\frac{d_{li}^k}{v_i} * x_{li}^k \leq T, \forall u_i \in U, l \in L, j_l^k \in J_l \tag{18}$$

$$x_{li}^k \in \{0,1\}, \forall u_i \in U, l \in L, j_l^k \in J_l \tag{19}$$

**Algorithm 3** MPF-RUA.

```
Require:
    $J$, $L$;
Ensure:
    $profit$, $x_l = \{x_{l1}^1, ..., x_{lm}^{n_l}\}$, $\forall l \in L$;
1:  Initialize $profit \leftarrow 0$, $x_l = \{0, ..., 0\}$, $\forall l \in L$; $U \leftarrow \phi$.
2:  for $l$ from 1 to $q$ do
3:      if $J_l \neq \emptyset$ then
4:          $J^* \leftarrow \emptyset$.
5:          Construct set $UJ_l$, which contains all user-task ($u_i$,$j_x$) pairs, user $u_i \in U$, $j_x \in J_l$.
6:          while $U \neq \emptyset$ && $J_l \neq \emptyset$ do
7:              Select a user-task pair ($u_{i^*}$, $j_l^k$) from $UJ_l$, which has the maximum profit.
8:              if $profit_{li^*}^k > 0$ then
9:                  Remove all user-task pairs containing either user $u_{i^*}$ or task $j_l^k$ from $UJ_l$.
10:                 $J^* \leftarrow J^* \cup j_l^k$.
11:                 $profit \leftarrow profit + profit_{li^*}^k$, $x_{li^*}^k \leftarrow 1$.
12:             else
13:                 Break.
14:         $J_l \leftarrow J_l \backslash J^*$.
15:
16:     if a user $u_i$ arrives at the platform then
17:         Find a feasible $(u_i, j_{l^*}^k)$, which has the maximum profit, $j_{l^*}^k \in J_{l-T+1} \cup J_{l-T+2} \cup ... \cup J_l$.
18:         if $j_{l^*}^k$ exists then
19:             $J_{l^*} \leftarrow J_{l^*} \backslash j_{l^*}^k$.
20:             $profit \leftarrow profit + profit_{l^*i}^k$, $x_{l^*i}^k \leftarrow 1$.
21:         $U \leftarrow U \cup u_i$.
22:
23:     if a user $u_i \in U$ has no sparse time for completing any task and further he still has task result to upload then
24:         Find the shortest $feasible$ detour route ($u_i$ -AP-destination) for data offloading, if applicable.
25:         $U \leftarrow U \backslash u_i$.
26:
27:     At the end of timeslot $l$, performing task-accomplishment-triggered data offloading (if applicable) by calling the lines 6-19 in BMA-RUA. //For checking at the end of timeslot $l$.
28:     for each user $u_i \in U$ do
29:         Upload task results in hand that have reached the upload deadline via cellular network.
30: return $profit$, $x_l = \{x_{l1}^1, ..., x_{lm}^{n_l}\}$, $\forall l \in L$.
```

**Algorithm 3** MPF-RUA

$$\phi_{li} \in \{0,1\}, \forall u_i \in U, l \in L \tag{20}$$

Constraint (13) guarantees that each user can only complete tasks which start after the user arrives at the platform. Constraint (14) guarantees that each user can only complete tasks which terminate before the user leaves the platform. Constraint (15) guarantees that each user can reach his destination in time after completing the assigned task(s). Constraints (16), (17), (18), and (19) correspond to constraints (8), (9), (10), and (11), respectively.

The task assignment problem in user-online-arriving scenario is also a 0–1 integer linear programming problem. Because users arrive and depart dynamically, the user-online-arriving task assignment problem is even more complicated compared with the user-offline-arriving scenario. To address this problem, we again seek optimal task assignment for each individual timeslot and accordingly design an online algorithm.

## 5.2 MPF-RUA algorithm

In this subsection, we design an online task assignment algorithm. This algorithm adopts greedy maximum-profit-first strategy for task assignment and reducing upload cost via detour based data offloading, if possible. To pursue maximum platform profit, MPF-RUA always assigns users with tasks which can bring the highest profit. Specifically, upon the arrival of a user at the platform, MPF-RUA will assign the user with a task which leads to the maximum profit; Upon task arrivals at the beginning of a slot, MPF-RUA will assign them to users, which bring the maximum profit, if possible. Before the assignment between a user and a task, the following task-undertaking eligibility conditions will be checked:

1) Whether the user can complete the task on time,
2) Whether the user can still reach his destination after finishing the task before given deadline, and
3) Whether the task assignment brings positive profit for the platform.

If any of the above conditions is violated, the task cannot be assigned to the user.

In MPF-RUA, the following events can be observed.

Task arrival event. We assume task arrivals occur at the beginning of timeslots. Since each task is required to be submitted to the platform $T$ timeslots before its actual start, for any timeslot $l$, the platform will have all the information of those task(s) supposed to start at the beginning of the $l+T$ timeslot.

User arrival event. User arrival can happen at any time in a timeslot. Upon the arrival of a user, he is eligible for undertaking a task immediately provided that all the task-undertaking eligibility conditions are met. The newly arriving user be recruited to undertake tasks that have arrived at the platform in previous timeslots but have not yet started.

User leaving event. After a user is scheduled to move all way to his destination, the user is considered as having left the system and will be removed from the user set for task assignment. In such movement, a detour route in the form of "user-AP-destination" for data offloading, if he still has task result to upload.

Data offloading event. After a user finishes a task, the user can utilize APs (if applicable) for data offloading. This event is similar with corresponding data offloading strategy in BMA-RUA. The user can take a detour for data offloading in the form of "user-AP-NextTaskLocation" or in the form of "user-AP".

Algorithm 3 gives the procedure of the MPF-RUA algorithm. Line 1 initializes necessary variables. Lines 2–29 describe the online task assignment process. Lines 3–14 describe how to handle task arrivals at the beginning of a slot. Line 7 selects a feasible user-task pair that leads to the maximum profit provided that all the task-undertaking eligibility conditions are met. Then lines 8–11 update corresponding variables. Lines 16–21 describe how to handle user arrival events. The tasks waiting for assignment at this moment include all those tasks that have arrived in the previous $T$ timeslots but not assigned yet. Lines 23–25 describe how to handle user departure. A user with no enough spare time for task execution means it's time for him to move all way to his destination. Line 27 describes how to handle data offloading event. Finally, Lines 28–29 check whether each task result reaches its upload deadline. If so, the task result must be uploaded to the platform via cellular network immediately.

The time complexity of MPF-RUA is deduced as follows. Assume the user arrival rate is $\lambda$ users/slot, and the average duration that a user stays in the system is $D$, we have the expected number of users in the system will be $m=\lambda D$. The size of set $UJ_l$ will be $O(mn)$, where $n$ is the average number of task arrivals in a slot. Suppose binary priority queue is used, Line 7 takes $O(\log(mn))$ time, and line 9 takes $O((m+n)\log(mn))$ time for the queue management. Then lines 7–13 take $O((m+n)\log(mn))$ time. Since the "while" loop in lines 6–13 can be iterated at most $O(\min\{m, n\})$ time, lines 6–13 thus take $O(\min\{m, n\}(m+n)\log(mn))$ time. Accordingly, lines 3–14 take $O(\min\{m, n\}(m+n)\log(mn))$ time. Lines 16–21 take $O(nT)$ time and lines 23–25 take $O(1)$ time. Line 27 takes $O(mw)$ time and lines 28–29 take $O(m)$ time. Therefore, the overall complexity of MPF-RUA will be $O(q\min\{m,n\}(m+n)\log(mn))$.

## 6 Performance evaluation

In this section, extensive simulations are conducted for evaluating the performance of our proposed algorithms. The simulator was developed using PYTHON and all the simulations were carried out on a desktop computer with Microsoft Windows 10, 8 GB memory, Intel Core i7-10700F CPU, and 2.90 GHz clock-speed.

## 6.1 Simulation settings

The sensing area used in the simulations is a 60×60 square region. The travel distance is measured by Euclidean distance. The total number $q$ of timeslots is set to 30. The length of a task's time window ranges from 1 to 4 timeslots. The reward for the accomplishment of a task as provided by requesters ranges from 3 to 10. The traveling speed of a user is randomly and uniformly chosen from 10–20/timeslot. The default number of APs is 6, each having a uniform coverage radius of 1. The default maximum upload deadline and default interval between task start and arrival are both set to 1 timeslot. The unit time upload cost (per timeslot) and the bonus for a user to complete a unit task are all set to 1. The unit travel distance cost is set to 0.2. The task arrivals at the platform follow normal distribution with mean value $\mu$ and standard variance $\sigma$. The standard variance $\sigma$ is set to 1 and the value of $\mu$ changes with user arrival scenario, which is set to 8 for the offline case and 5 for the online case. The tasks and APs are spatially uniformly distributed in the sensing map. The above simulation settings are similar to those used in [21, 22]. For each parameter setting, 20 independent experiments were conducted and the average results were reported. Table 2 shows the default parameter settings used in the simulations.

The user setting for different scenario is shown as follows:

1) User-offline-arriving scenario. There are totally 20 mobile users. The initial locations of users are randomly and uniformly chosen in the simulation map at the beginning.
2) User-online-arriving scenario. Arrivals of users follow Poisson distribution with a default user arrival rate 4 users/slot. The start locations and destinations of users are randomly and uniformly chosen from the sensing map. The total time that a user can stay in the system (i.e., starting from the timeslot when he arrives at the system to the timeslot he must reach his destination before pre-determined deadline) ranges from 4 to 12 timeslots.

For each scenario, we simulated different algorithms and verified multiple performance measures.

## 6.2 Simulation results for user-offline-arriving scenario

For the offline scenario, we simulated the following four algorithms.

- Benchmark algorithm (Bench), proposed in this paper.
- BMA-RUA, proposed in this paper.
- BMA: In this algorithm, the platform performs BMA-RUA for task assignment but without implementing the detour based data offloading in BMA-RUA. This algorithm was proposed in [22].
- DSF-offline (Distance-Shortest-First-offline): In this algorithm, the platform always selects the user-task pair, which leads to the shortest distance, one task for each time. This process continues until no available pair exists.

**Table 2** Default parameter settings

| Parameters | Values |
|---|---|
| Number of users (for offline scenario) | 20 |
| User moving speed | 10–20/timeslot |
| Total number of timeslots $q$ | 30 |
| Number of APs | 6 |
| AP coverage radius | 1 |
| Task time window length | 1–4 timeslots |
| Unit travel cost $K_d$ | 0.2 |
| Unit upload cost $C$ | 1 |
| Unit task bonus $K_t$ | 1 |
| Data amount $D_t$ generated per unit time | 1 |
| Task reward | 3–10 |
| Maximum upload deadline $K$ | 1 timeslot |
| Interval between task start and arrival $T$ | 1 timeslot |
| Task arrival distribution | $N(8, 1)$ for offline scenario; $N(5, 1)$ for online scenario |

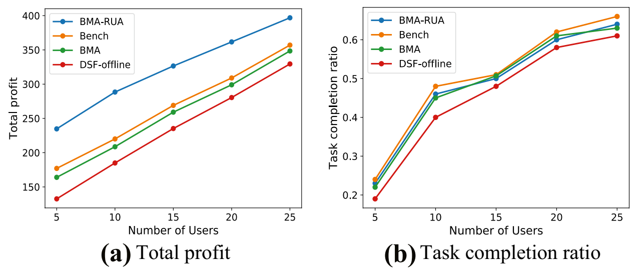


**Fig. 4** Impact of number of users

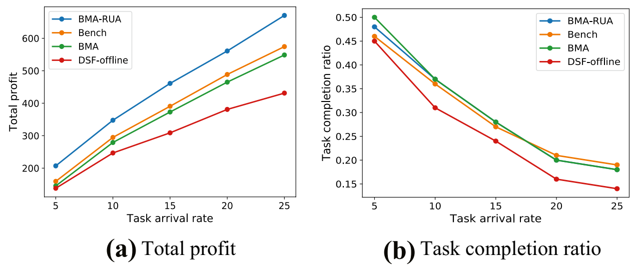


**Fig. 5** Impact of task arrival rate

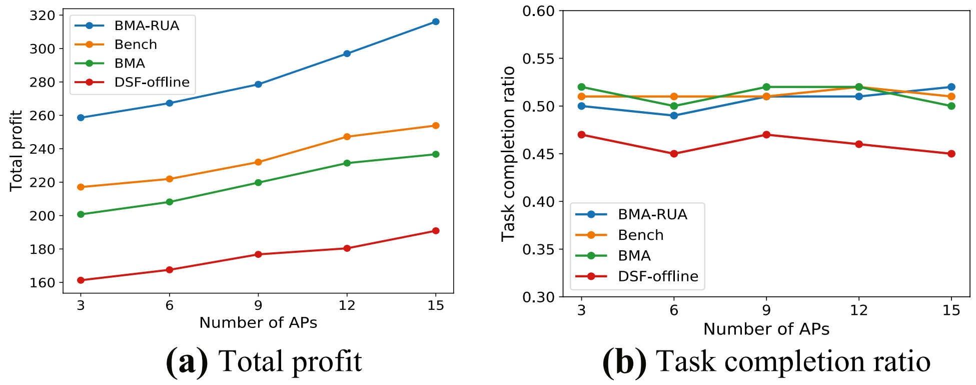


**Fig. 6** Impact of number of APs

Figure 4 compares the performance by different algorithms versus number of users. In Fig. 4a, we can see that the total profits by all algorithms increase as the number of users increases. This is because more users can perform more tasks and further reduce the travel costs for task executions. The rank from the best to the worst is as follows: BMA-RUA, Bench, BMA, and DSF-offline. The curve by BMA is close to that by Bench. BMA-RUA performs the best among all the algorithms due to the introduction of detour based data offloading. In Fig. 4b, the task completion ratios of all algorithms increase as the number of users increases. This is because the increase of number of users can lead to completion of more tasks. Also, also it can be seen that the bipartite graph based methods (i.e., BMA-RUA, Bench, and BMA) can achieve higher task completion ratio compared with greedy method (i.e., DSF-offline).

Figure 5 compares the performance by different algorithms versus task arrival rate. In Fig. 5a, it is seen that the total profits by all the algorithms increase with the task arrival rate increasing. Again, the rank from the best to the worst is the same to that in Fig. 4. It can also be seen that when the task arrival rate is low, the profits by all the algorithms (except BMA-RUA) are quite close. This is because lower task arrival rate leads to less competition among the task assignments and thus similar results by the algorithms. Also, we can again see that BMA-RUA performs the best among all the simulated algorithms. In Fig. 5b, the task completion ratios of all algorithms decrease as the task arrival rate increases. This is because the number of users is limited and thus the increase of number of tasks causes reduction of the task completion rate.

Figure 6 compares the performance by different algorithms versus number of APs. In Fig. 6a, the same rank of algorithms is observed. Also, it is seen that the profit growth rate by BMA-RUA is higher than that by other algorithms. This is because more APs lead to significantly increased offloading opportunities due to the use of detours based data offloading in BMA-RUA. In Fig. 6b, the task completion ratios of all algorithms are basically the same as the number of APs varies. This is because the increase of number of APs has little effect on the task completion rate.

Figure 7 shows the impact of $T$ on total profit of BMA-RUA. In this experiment, $T$ ranges from 1 to 7 timeslots. The total profit by BMA-RUA first increases and then gradually stabilizes as $T$ is beyond certain threshold. This is because when $T$ is small, increased $T$ allows more flexibility on task assignment. However, when $T$ is beyond certain threshold, the performance will not increase any further. The reason is as follows. Based on our setting of $\mu = 8$ and task load randomly chosen in the range [1, 3] timeslots, we have the average number of tasks in the system will be $8 \times (1+3)/2 = 16$ while there are in total 20 fully available users in the systems. Thus, further increasing of $T$ will not lead to further increase of performance.

Figure 8 shows the impact of maximum upload deadline on total profit by BMA-RUA. It is seen that, with the increase of maximum upload deadline, the total profit by BMA-RUA also increases. This is because users will have more available time for data offloading via longer detours with maximum deadline increasing.

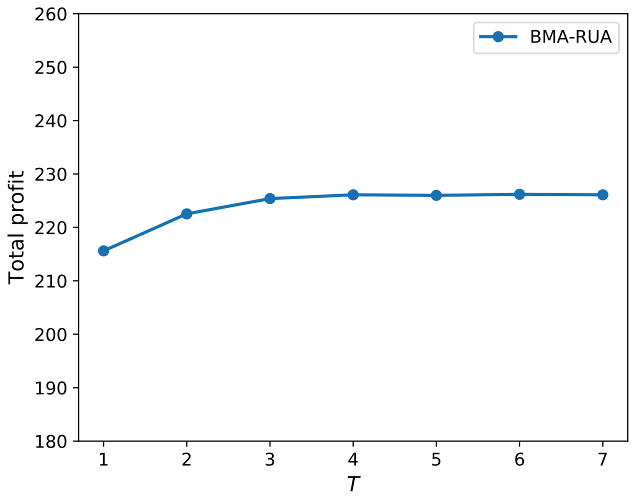


**Fig. 7** Total profit by BMA-RUA with varying $T$

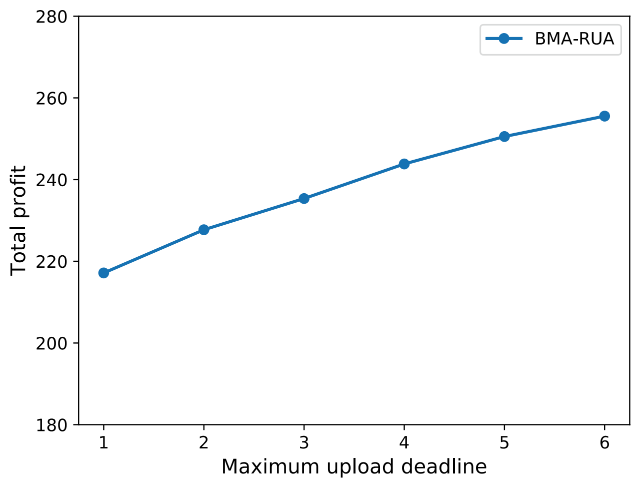


**Fig. 8** Total profit by BMA-RUA with varying maximum upload deadline

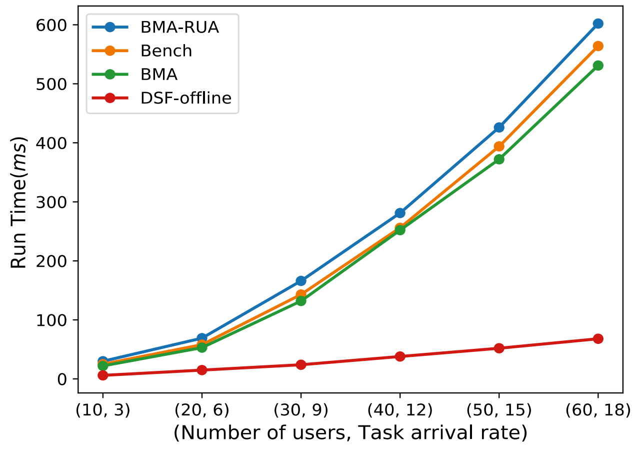


**Fig. 9** Runtime by different algorithms with varying number of users and task arrival rate

Figure 9 shows the runtime by different algorithms with varying combination of number of users and task arrival rate. In this figure, the rank from the longest to the shortest in terms of runtime is as follows: BMA-RUA, Bench, BMA, and DSF-offline. BMA-RUA, Bench, and BMA have faster growth rates, while DSF-offline has a slower growth rate. This is because the bipartite graph matching-based strategy adopted by the first three algorithms takes more time than the greedy strategy adopted by DSF-offline.

## 6.3 Simulation results for user-online-arriving scenario

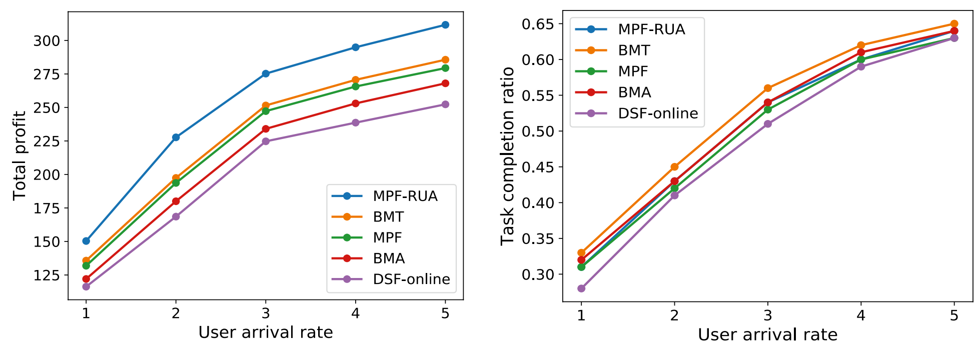
For the scenario, we compare the following algorithms:

- MPF-RUA, proposed in this paper.
- MPF: In this algorithm, the platform performs MPF-RUA for task assignment but without implementing the detour based data offloading in MPF-RUA. In this way, we can observe the benefit brought by the detour based data offloading in MPF-RUA.
- DSF-online (Distance-Shortest-First-online): In this algorithm, each time the user-task pair leading to the shortest distance is always chosen. This process continues until no such pair exist.
- BMA: In this algorithm, task assignments were carried out at the beginning of each timeslot upon task arrivals (by using bipartite-graph-matching algorithm) but no task assignment is done in the middle of timeslots upon user arrivals. Moreover, no data offloading is performed in this algorithm.
- BMT: This algorithm works similarly to MPF for task assignment but uses optimal per-slot bipartite-graph matching algorithm for the task assignment at the beginning of each slot rather than using the maximum-profit-first strategy like in MPF.

Figure 10 compares the performance by different algorithms versus user arrival rate. In Fig. 10a, we can see that the total profits by all the algorithms increase with the user arrival rate. However, the profit increase rate starts dropping after the user arrival rate is beyond 3 user arrivals per slot. This is because in this case some already arrived users will not be assigned any task due to the assumption of fixed task arrival rate. It is also seen that the rank from the best to the worst is as follows: MPF-RUA, BMT, MPF, BMA, DSF-online. The curve by MPF is very close to that by BMT while much better than that by BMA. The former means the performance of the maximum-profit-first strategy in MPF is very close to that of optimal per-slot matching algorithm and the latter means the user-arrival-triggered task assignment is quite efficient. Moreover, the allowance of detour-based data offloading in MPF-RUA achieves significantly increased profit as compared with other algorithms. In Fig. 10b, it can be seen that the completion ratio of all algorithms increase as the number of users increase. This is because more users can lead to more tasks be finished.

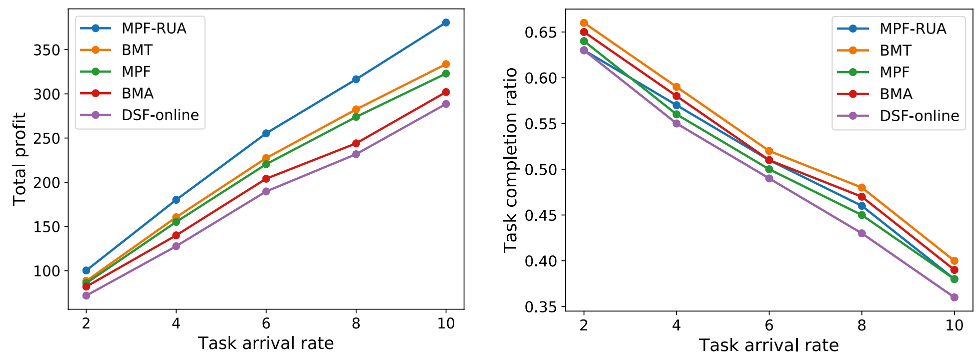
Figure 11 compares the performance by different algorithms versus task arrival rate. In Fig. 11a, we can see that the total profits by all the algorithms increase as the task arrival rate increases. This is because more task arrivals can lead to high profits for those tasks assigned to users. It is also seen when the task arrival rate is low, the total profits by all the algorithms except MPF-RUA are quite close. This is because each task in this case can be assigned to the most profitable user no matter which algorithm is used. In Fig. 11b, the completion ratios of all algorithms decease as the task arrival rate increases. The reason is similar to that for Fig. 5b.

**Fig. 10** Impact of user arrival rate



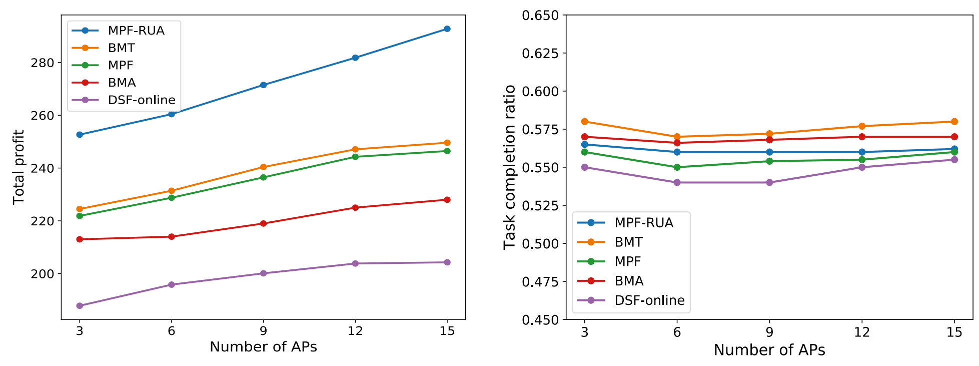
**(a)** Total profit **(b)** Task completion ratio

**Fig. 11** Impact of task arrival rate



**(a)** Total profit **(b)** Task completion ratio

**Fig. 12** Impact of number of APs



**(a)** Total profit **(b)** Task completion ratio

Figure 12 compares the performance by different algorithms versus number of APs. In Fig. 12a, the rank from the best to the worst in terms of total profit is as follows: MPF-RUA, BMT, MPF, BMA, and DSF-online. Also, it is seen that MPF-RUA has a stable profit growth rate as the number of APs increases. This is because that more APs can bring more detour-based data offloading opportunities. In Fig. 12b, the task completion ratios of all algorithms are basically the same as the number of APs varies. The reason is also similar to that for Fig. 6b.

Figure 13 shows the effect of *T* on total profit by MPF-RUA. In this figure, it is seen that the total profit by MPF-RUA first increases and then gradually stabilizes as *T* keeps increasing, which is similar to that by BMA-RUA (see Fig. 7). This is because the number of users in this test is limited, which restricts the upper bound of the total profit that can be obtained.

Figure 14 shows the effect of maximum upload deadline on the total profit by MPF-RUA. In this figure, we can see that, with the increase of maximum upload deadline, the

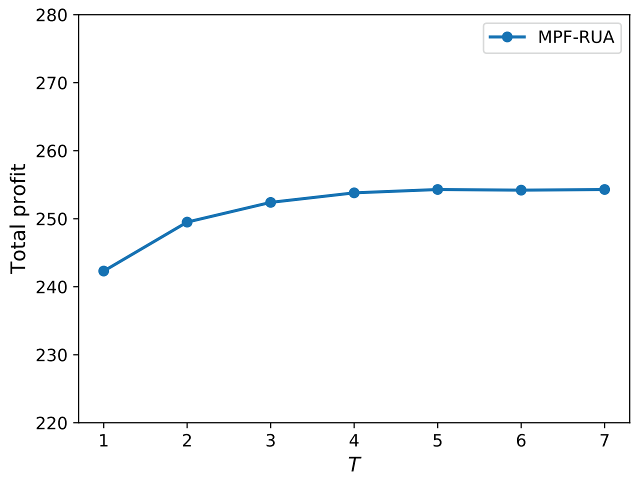


**Fig. 13** Total profit by MPF-RUA with varying $T$

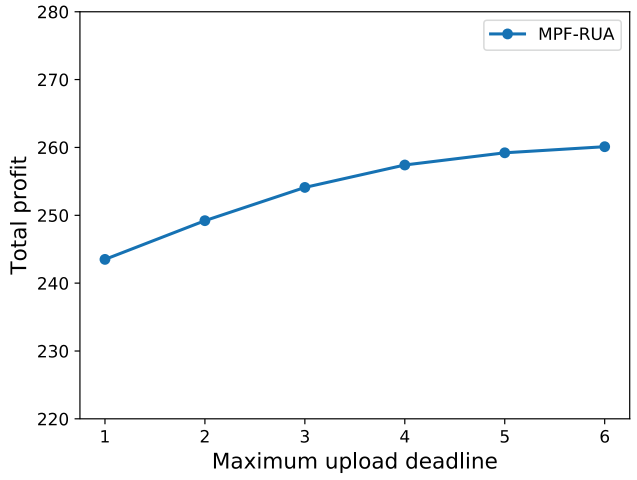


**Fig. 14** Total profit by MPF-RUA with varying maximum upload deadline

performance of MPF-RUA also increases, which is similar to that of BMA-RUA (see Fig. 8).

Figure 15 shows the runtime by different algorithms with varying combination of user arrival rate and task arrival rate. In this figure, the rank from the longest to the shortest in terms of runtime is as follows: BMT, BMA, MPF-RUA, MPF, and DSF-online. Since both BMT and BMA adopt the bipartite graph matching-based strategy for task assignment, they take more runtime than the other three algorithms for task assignment.

## 7 Conclusion and future work

In this paper, we studied time sensitive task assignment in participatory sensing. We assume that tasks arrive dynamically and each task is associated with a specific time window for

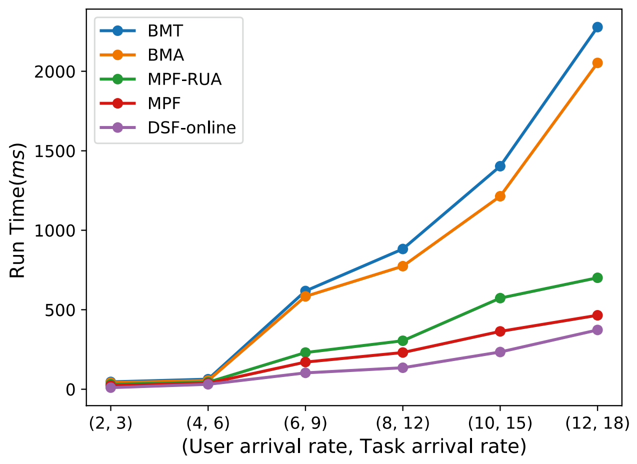


**Fig. 15** Runtime by different algorithms with varying user arrival rate and task arrival rate

task execution. We formulated the profit maximization problems for user-offline-arriving-scenario and user-online-arriving scenario. For the user-offline-arriving scenario, we designed a benchmark algorithm and an online algorithm, which adopts bipartite-matching-based strategy for task assignment and further performs minimal detour based data offloading for reducing the data upload cost, whenever possible. For the user-online-arriving scenario, we designed an online algorithm, which adopts a maximum-profit-first strategy for task assignment and also minimal detour based data offloading for reduction of data upload cost whenever applicable. For each of the algorithms, detailed design was presented and computation complexity was deduced. Extensive simulations were conducted and the simulation results verified the effectiveness of our proposed algorithms.

There are some research directions for future study in the direction of time-window based task assignment for mobile crowdsensing. First, design of effective incentive mechanisms can effectively improve the task assignment performance while respecting the rationality of various participants. In this aspect, it is interesting to design effective auction mechanism or pricing strategy to incentivize the involvement of various participants in the system while maintaining high task assignment performance. Second, the willingness of users for task executions is also an important factor affecting the task assignment performance, which deserves further study.

## Appendix

Here, we describe how to find the point $X$ leading to the minimal length of path $A$–$X$-$B$ to resolve the pilgrimage to castrum problem.

A Cartesian coordinate system is first constructed (see Fig. 16). Given the user's initial location (denoted by $A$), the

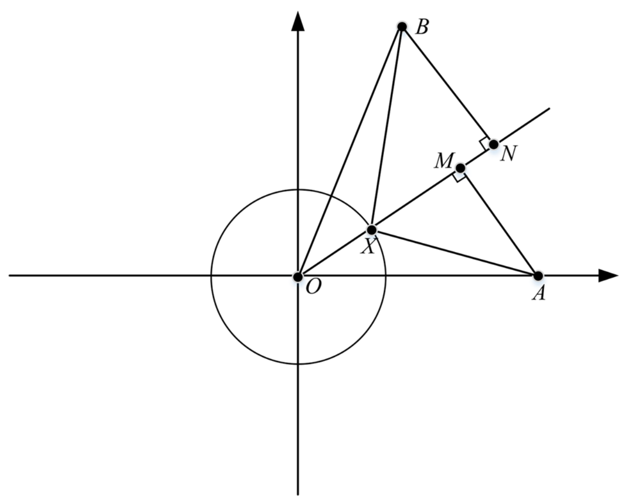


**Fig. 16** Illustration for finding point $X$ leading to minimal length of path $A$–$X$-$B$. In this figure, $AM \perp OX$ and $BN \perp OX$

user's target location (denoted by $B$), and the circular castrum, which is centered at $O$ and has a radius $r$, the problem is to find the point $X$ leading to the minimal length of path $A$–$X$-$B$. Denote the coordinate of $A$ as $(x_A, 0)$, the coordinate of $B$ as $(x_B, y_B)$. Denote $\angle AOX$ as $\theta$, $\angle AOB$ as $\alpha$. Then we have the coordinate of point $X$ as $(r \cdot cos\theta, r \cdot sin\theta)$.

By using geometric methods, we can find the point $X$ which leads to the minimal distance of path $A$–$X$-$B$ satisfies $\angle AXM = \angle BXN$. Then we have $tan\angle AXM = tan\angle BXN$. So we have:

$$\frac{x_A cos\theta - r}{x_A sin\theta} = \frac{\sqrt{x_B^2 + y_B^2}\cos(\alpha - \theta) - r}{\sqrt{x_B^2 + y_B^2}\sin(\alpha - \theta)}. \tag{21}$$

Since $cos(\alpha\text{-}\theta) = cos\alpha \cdot cos\theta + sin\alpha \cdot sin\theta$ and $sin(\alpha\text{-}\theta) = sin\alpha \cdot cos\theta — cos\alpha \cdot sin\theta$, we have:

$$\frac{x_A cos\theta - r}{x_A sin\theta} = \frac{x_B cos\theta + y_B sin\theta - r}{y_B cos\theta - x_B sin\theta} \tag{22}$$

Denote $tan\frac{\theta}{2} = x$, then we have:

$$sin\theta = \frac{2x}{1 + x^2} \tag{23}$$

$$cos\theta = \frac{1 - x^2}{1 + x^2} \tag{24}$$

Combine Eqs. (22), (23) and (24) together, we have:

$$\begin{aligned}&\left(x_A + r\right)y_B x^4 + \left[4x_A x_B + 2r(x_A + x_B)\right]x^3 - 6x_A y_B x^2 \\ &+ \left[2r\left(x_A + x_B\right) - 4x_A x_B\right]x + \left(x_A - r\right)y_B = 0\end{aligned} \tag{25}$$

Equation (25) is a quartic equation with unknown quantity $x$. By using some math tools (such as Matlab), the equation can be easily solved. Then we can get the value of $\theta$ (i.e., $\angle AOX$). Therefore, the coordinate of point $X$ can be obtained.

**Author contribution** Shuo Peng formulated the problems, designed the algorithms, did the simulation coding and debugging job, and prepared the original draft. Kun Liu involved part of the algorithms design and complexity deduction. Shiji Wang involved the writing, review, editing, and project administration. Yangxia Xiang involved the simulation software development and validation. Baoxian Zhang supervised this work, involved the writing, review, editing, and also funding acquisition. Cheng Li involved the writing, review, editing, and also funding acquisition.

**Funding** This work was supported in part by the NSF of China under Grant No. 61872331, the Natural Sciences and Engineering Research Council (NSERC) of Canada (Discovery Grant RGPIN-2018–03792), and the InnovateNL SensorTECH Grant 5404–2061-101.

**Data availability** Non Applicable.

**Code availability** Available from the authors upon request.

## Declarations

**Competing interests** The authors declare no competing interests.

**Ethics approval** This work does not involve any work related to ethics.

**Consent to publish** All authors consent to publication.

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Guo B, Wang Z, Yu Z, Wang Y, Yen NY, Huang R, Zhou X (2015) Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm. ACM Comput Surv 48(1):1–31
2. Liu Y, Kong L, Chen G (2019) Data-Oriented Mobile : A Comprehensive Survey. IEEE Commun Surv Tutorials 21(3):2849–2885 (third quarter)
3. S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher (2015) SmartRoad: Smartphone-Based Crowd Sensing for Traffic Regulator Detection and Identification. ACM TOSN 11(4):1–27
4. Dutta J, Gazi F, Roy S, Chowdhury C (2016) AirSense: Opportunistic crowd-sensing based air quality monitoring system for smart city, in Proc. IEEE Sensors 2016, pp. 1–3
5. Zheng Y, Liu F, Hsieh H (2013) U-Air: When urban air quality inference meets big data, in Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, pp. 1436–1444
6. Aram S, Troiano A, Pasero E (2012) Environment sensing using smartphone, in Proc. IEEE Sensors Applications Symposium, pp. 1–4
7. Yang Z, Shangguan L, Gu W, Zhou Z, Wu C, Liu Y (2014) Sherlock: Micro-Environment Sensing for Smartphones. IEEE Trans Parallel Distrib Syst 25(12):3295–3305

8. Wu C, Yang Z, Liu Y (2015) Smartphones based crowdsourcing for indoor localization. IEEE Trans Mob Comput 14(2):444–457
9. Gong W, Zhang B, Li C (2018) Task Assignment in Mobile Crowdsensing: Present and Future Directions. IEEE Network 32(4):100–107
10. Peng S, Zhang B, Yan Y, Li C (2021) Time Window-based Online Task Assignment for Mobile Crowdsensing, in Proc. of IEEE ICC 2021, pp. 1–6
11. He S, Shin D, Zhang J, Chen J (2014) Toward optimal allocation of location dependent tasks in crowdsensing, in Proc. of IEEE INFOCOM 2014, pp. 745–753
12. Wang X, Jia R, Tian X, Gan X (2018) Dynamic task assignment in crowdsensing with location awareness and location diversity, in Proc. IEEE INFOCOM 2018, pp. 2420–2428
13. Li H, Li T, Wang W, Wang Y (2019) Dynamic Participant Selection for Large-Scale Mobile Crowd Sensing. IEEE Trans Mob Comput 18(12):2842–2855
14. Wang J, Wang F, Wang Y, Wang L, Qiu Z, Zhang D, Guo B, Lv Q (2020) HyTasker: Hybrid Task Allocation in Mobile Crowd Sensing. IEEE Trans Mob Comput 19(3):598–611
15. Liu Y, Guo B, Chen C, Du H, Yu Z, Zhang D, Ma H (2019) FooDNet: Toward an Optimized Food Delivery Network Based on Spatial Crowdsourcing. IEEE Trans Mob Comput 18(6):1288–1301
16. Yang Y, Liu W, Wang E, Wu J (2019) A Prediction-Based User Selection Framework for Heterogeneous Mobile CrowdSensing. IEEE Trans Mob Comput 18(11):2460–2473
17. Yucel F, Yuksel M, Bulut E (2021) Coverage-aware Stable Task Assignment in Opportunistic Mobile Crowdsensing. IEEE Trans Veh Technol 70(4):3831–3845
18. Wang L, Yu Z, Han Q, Guo B, Xiong H (2018) Multi-Objective Optimization Based Allocation of Heterogeneous Spatial Crowdsourcing Tasks. IEEE Trans Mob Comput 17(7):1637–1650
19. Kang Y, Miao X, Liu K, Chen L, Liu Y (2015) Quality-aware online task assignment in mobile crowdsourcing, in Proceedings of IEEE MASS 2015, pp. 127–135
20. 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
21. Li X, Zhang X (2021) Multi-Task Allocation Under Time Constraints in Mobile Crowdsensing. IEEE Trans Mob Comput 20(4):1494–1510
22. Tao X, Song W (2021) Profit-Oriented Task Allocation for Mobile Crowdsensing with Worker Dynamics: Cooperative Offline Solution and Predictive Online Solution. IEEE Trans Mob Comput 20(8):2637–2653
23. Xu J, Xiang J, Yang D (2015) Incentive Mechanisms for Time Window Dependent Tasks in Mobile Crowdsensing. IEEE Trans Wireless Commun 14(11):6353–6364
24 Xu J, Fu J, Yang D, Xu L, Wang L, Li T (2017) FIMI: A Constant Frugal Incentive Mechanism for Time Window Coverage in Mobile Crowdsensing. J Comput Sci Technol 32(5):919–935
25. Sun X, Yang X, Wang C, Wang J (2020) A Novel User Selection Strategy with Incentive Mechanism Based on Time Window in Mobile Crowdsensing. Discret Dyn Nat Soc. Article ID 2815073, 13. https://doi.org/10.1155/2020/2815073
26. Hu T, Xiao M, Hu C, Gao G, Wang B (2017) A QoS-sensitive task assignment algorithm for mobile crowdsensing. Pervasive Mobile Comput 41:333–342
27. Tao X, Song W (2019) Location-Dependent Task Allocation for Mobile Crowdsensing With Clustering Effect. IEEE Internet Things J 6(1):1029–1045
28. Liu Y, Guo B, Wang Y, Wu W, Yu Z, Zhang D (2016) TaskMe: multi-task allocation in mobile crowd sensing, in Proc. of ACM UbiComp 2016, pp. 403–414
29. Peng S, Gong W, Zhang B, Zhao Y, Li C (2020) AP-Assisted Online Task Assignment for Mobile Crowdsensing. Mob Netw Appl 25(5):1694–1707
30. 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
31 Wang E, Yang Y, Wu J, Liu W, Wang X (2018) An Efficient Prediction-Based User Recruitment for Mobile Crowdsensing. IEEE Trans Mobile Comput 17(1):16–28
32. Munkres J (1957) Algorithms for the assignment and transportation problems. Soc Indust Appl Math 5(1):32–38

**Shuo Peng** received the B. Eng. degree from Civil Aviation University of China, Tianjin, China, in 2017. He is currently working towards the Ph. D. degree in computer science at the University ofChinese Academy of Sciences, Beijing, China. His research interests include mobile crowdsensing and Internet of Things. Mailing address: Research Center of Ubiquitous Sensor Networks, University of the Chinese Academy of Sciences, 19A Yuquan Road, Beijing 100049, China. Email: pengshuo17@mails.ucas.ac.cn.

**Kun Liu** received the B. Eng. degree from Henan University, Kaifeng, China, in 2018. He is currently pursuing the Ph. D. degree in computer science at the University of Chinese Academy of Sciences, Beijing, China. His research interests include mobile crowdsensing and Internet of Things. Mailing address: Research Center of Ubiquitous Sensor Networks, University of the Chinese Academy of Sciences, 19A Yuquan Road, Beijing 100049, China. Email: liukun181@mails.ucas.edu.cn.

**Shiji Wang** received the Ph.D. degrees from Harbin Institute of Technology (HIT), Harbin, China, in 2006. Since 2007, he has worked in Beijing Aerospace Measurement and Control Technology Co., Ltd., mainly engaged in the research of high-end electronic measurement instrument technology. Now he is a researcher and chief engineer of Beijing Aerospace TT & C Technology Co., Ltd. His research interests include basic measurement instruments, communication test instruments, RF/microwave test instruments, broadband communication and advanced bus technology. He has 62 individual authorized patents, and published more than 50 papers. He led the team to apply for more than 50 industrialization and pre research projects related to electronic measuring instruments, such as major instrument projects of the Ministry of science and technology, with a total scientific research fund of over 200 million yuan. He developed many high-end electronic measuring instruments such as 20GSa/s oscilloscope, GHz vector signal generator and analyzer, and many technologies have reached the international advanced level. He won the national key new product certificate, international invention Silver Award, space Fund Award, space defense award, space outstanding contribution award, national defense science and technology progress award, etc. In 2016, he was selected into Beijing Science and technology new star talent plan and academic leader of the group for many years. Mailing Address: Beijing Aerospace Measurement and Control Technology Co.,Ltd. 100041. Email: wsj978418128@163.com.

**Yangxia Xiang** obtained the M. Eng. Degree in computer science from Army Engineering University of PLA, China, in 2004. Sheis currently working with Information and Communication Department, Army Academy of Armored Forces, Beijing, China. Mailing Address: No.21 Dujiakan Street,Fengtai District, Beijing 100072, China. Email: 18701539583@163.com.

**Baoxian Zhang** received his B.Sc., M.Sc., and Ph.D. degrees in Electrical Engineering from Northern Jiaotong University (now Beijing Jiaotong University), China, in 1994, 1997, and 2000, respectively. He is currently a Full Professor with the Research Center of Ubiquitous Sensor Networks at the University of Chinese Academy of Sciences, Beijing, China. He is an Associate Editor of IEEE Systems Journal. He has served as a co-chair for various technical symposia of several international conferences, including IEEE GLOBECOM, ICC,, and VTC. He has published over 200 refereed technical papers in archival journals and conference proceedings. His research interests cover network protocol and algorithm design, wireless networks, Internet of Things, and edge computing. Mailing address: Research Center of Ubiquitous Sensor Networks,University of the Chinese Academy of Sciences, 19A Yuquan Road, Beijing 100049, China. Email: bxzhang@ucas.ac.cn.

**Cheng Li** received the B.Eng. and M. Eng. degrees from Harbin Institute of Technology, Harbin, P. R. China, in 1992 and 1995, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Memorial University, St. John's, Canada, in 2004. He is currently a Full Professor at the Faculty of Engineering and Applied Science of Memorial University, St. John's, Canada. His research interests include mobile ad hoc and wireless sensor networks, wireless communications and mobile computing, switching and routing, and broadband communication networks. He is an associate editor of the IEEE Transactions on Communications, IEEE Internet-of-Things Journal, IEEE Network Magazine, and IEEE Systems Journal. Mailing address: Faculty of Engineering and Applied Science, Memorial University of Newfoundland St. John's, NL A1B 3X5, Canada. Email: licheng@mun.ca.