Online Task Assignment for Crowdsensing in Predictable Mobile Social Networks

Mingjun Xiao, *Member, IEEE*, Jie Wu, *Fellow, IEEE*, Liusheng Huang, *Member, IEEE*, Ruhong Cheng, and Yunsheng Wang, *Member, IEEE*

Abstract—Mobile crowdsensing is a new paradigm in which a crowd of mobile users exploit their carried smart phones to conduct complex sensing tasks. In this paper, we focus on the makespan sensitive task assignment problems for the crowdsensing in mobile social networks, where the mobility model is predicable, and the time of sending tasks and recycling results is non-negligible. To solve the problems, we propose an Average makespan sensitive Online Task Assignment (AOTA) algorithm and a Largest makespan sensitive Online Task Assignment (LOTA) algorithm. In AOTA and LOTA, the online task assignments are viewed as multiple rounds of virtual offline task assignments. Moreover, a greedy strategy of small-task-first-assignment and earliest-idle-user-receive-task is adopted for each round of virtual offline task assignment in AOTA, while the greedy strategy of large-task-first-assignment and earliest-idle-user-receive-task is adopted for the virtual offline task assignments in LOTA. Based on the two greedy strategies, both AOTA and LOTA can achieve nearly optimal online decision performances. We prove this and give the competitive ratios of the two algorithms. In addition, we also demonstrate the significant performance of the two algorithms through extensive simulations, based on four real MSN traces and a synthetic MSN trace.

Index Terms—Crowdsensing, delay tolerant network, mobile social network, online task assignment

1 Introduction

With the explosive increase of smartphones and iPads, crowdsensing has become an appealing paradigm for collecting sensing data over urban environments. This paradigm involves a crowd of mobile users that exploit the sensors embedded in their carried smart phones or iPads, such as GPS, camera, accelerometer, digital compass, and so on, to cooperatively perform some mobile sensing tasks [2]. Since the mobile users can conduct sensing tasks in a large-scale urban area without deploying extra devices, crowdsensing has stimulated a number of attractive applications, such as urban WiFi characterization, traffic information collection, map labeling, and so on [3], [4], [5], [6], [7]. So far, some frameworks, incentive mechanisms, and task assignment schemes have been designed for crowdsensing, such as in [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20].

In this paper, we focus on the task assignment problem in the crowdsensing based on Mobile Social Networks (MSNs). Consider a crowdsensing example, as shown in Fig. 1. A requester, called Alice, intends to collect the information about air quality, traffic flow, and WiFi signal strength at different time in some urban areas. The whole information collection is divided into many sensing tasks according to type, time, and area. The sensing results of air quality, traffic flow, and WiFi characterization can be recorded by using the photo, video, and data files, respectively. In this crowdsensing, Alice first assigns the sensing tasks to some MSN users, such as Bob and David. After having performed sensing tasks, these users will send the results back to Alice. Since sending tasks and returning results involve large-size data transmissions, they adopt the short-distance wireless communication technologies, instead of 3G/4G, so as to save the communication costs. For example, Alice and Bob encounter frequently, so that they can communicate with each other via Bluetooth (called *direct encounter*). Alice and David might encounter never, but they can communicate with each other via WiFi networks when they visit access points, respectively (called *indirect encounter*). In such a crowdsensing system, an important problem is how the requester assigns the tasks to other mobile users, so as to minimize the average makespan of different types of sensing tasks or minimize the largest makespan of the same type of sensing tasks.

In the above problems, the makespan of a task not only includes the time of this task being conducted by a mobile user, but also contains the time of the task being sent from the requester to this user and the time of the result being sent back to the requester. Nevertheless, the task can be sent and the result can be returned, only when the requester meets the mobile user directly or indirectly. In fact, due to the mobility of users, it will take some time for the requester

- *M. Xiao, L. Huang, and R. Cheng are with the School of Computer Science and Technology/Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei, Anhui 230026, P.R. China. E-mail: {xiaomj, lshuang}@ustc.edu.cn, cgruhg@mail.ustc.edu.cn.*
- *J. Wu is with the Department of Computer and Information Sciences, Temple University, 1805 N. Broad Street, Philadelphia, PA 19122. E-mail: jiewu@temple.edu.*
- *Y. Wang is with the Department of Computer Science, Kettering University, 1700 University Ave., Flint, MI 48504. E-mail: ywang@kettering.edu.*

Manuscript received 7 Apr. 2016; revised 22 Aug. 2016; accepted 5 Oct. 2016. Date of publication 11 Oct. 2016; date of current version 27 June 2017. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2016.2616473

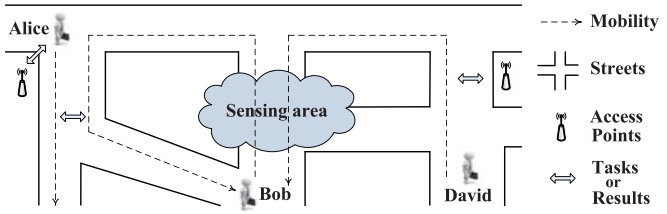


Fig. 1. Mobile crowdsensing in MSNs: Alice assigns tasks to and collects the results from other mobile users via Bluetooth when they meet, or via WiFi networks when they visit some connected access points.

to meet other users. The time of sending tasks and results is not negligible, so it needs to be counted into the makespan. Such characteristics make our problem different from previous task assignment problems in crowdsensing, such as [1], [10], [11], [12], [18], [20], where tasks and results are sent via cellular networks immediately, so that the corresponding time can be ignored.

Moreover, in the above problems, we consider two kinds of mobile sensing tasks: independent sensing tasks and collaborative sensing tasks. For independent sensing tasks, the makespan of each task is independent from others. We need to minimize the makespans of all tasks. Then, we focus on the Minimum-Average-Makespan (MAM) task assignment problem for this case. For collaborative sensing tasks, the whole sensing results can make sense only after all tasks have been completed. Since the makespan of the whole collaborative sensing tasks is the largest makespan of all tasks, we focus on the Minimum-Largest-Makespan (MLM) task assignment problem for this case.

To solve the MAM and MLM task assignment problems in mobile crowdsensing, we propose two online task assignment algorithms: the Average makespan sensitive Online Task Assignment (AOTA) algorithm and the Largest makespan sensitive Online Task Assignment (LOTA) algorithm. In AOTA and LOTA, the online task assignments are viewed as multiple rounds of virtual offline task assignments to be tackled with two greedy strategies. More specifically, our major contributions include:

1) We propose and formalize the makespan-sensitive task assignment problems for crowdsensing based on MSNs. Unlike existing crowdsensing task assignment problems, our problems take into consideration the time of sending tasks and results between mobile users, which is subjective to users' mobility.
2) We propose the online algorithm AOTA for the MAM task assignment problem. This algorithm is based on the greedy strategy of Small-Task-First-Assignment (STFA) and Earliest-idle-User-Receive-Task (EURT), in which the requester assigns the task with the smallest workload, one-by-one, to the earliest idle mobile user (i.e., the user who finished the tasks in hand earliest). Moreover, we analyze the estimation error and the competitive ratio of the AOTA algorithm.
3) We also propose the online algorithm LOTA for the MLM task assignment problem. The LOTA algorithm is based on the greedy strategy of Large-Task-First-Assignment (LTFA) and Earliest-idle-User-Receive-Task (EURT), in which the requester assigns the task with the largest workload, in turn, to the earliest idle mobile user. Furthermore, we also analyze the estimation error and the competitive ratio of this algorithm.
4) We conduct extensive simulations on four real traces and a synthetic trace to evaluate the proposed online algorithms. The results show that the proposed algorithms can achieve better performances on the average makespan and the largest makespan than other compared algorithms, respectively.

The remainder of the paper is organized as follows. We introduce the crowdsensing model and the problem in Section 2. The AOTA and LOTA algorithms are proposed in Sections 3 and 4, respectively. In Section 5, we evaluate the performance of our algorithms through extensive simulations. After reviewing related work in Section 6, we conclude the paper in Section 7.

2 Model & Problem

In this section, we introduce the crowdsensing model, followed by the problem definition.

2.1 Model

We consider an MSN that is composed of a crowd of mobile users, denoted by the set $V = \{v_0, v_1, \ldots, v_n\}$. Suppose that there is a user in this MSN, called the *requester*, who has some indivisible mobile sensing tasks. However, the total workload of these tasks is beyond its processing ability. Then, it starts crowdsensing. Other users in this MSN are assumed to be willing to participate in this crowdsensing due to some incentive mechanisms, such as [9]. Nevertheless, when these users conduct the crowdsensing tasks, they prefer to adopt the short-distance wireless communication model, so as to save the communication costs. More specifically, the requester moves around. If it encounters another mobile user, it assigns one or more tasks to this user. Then, this user will process these tasks. This might take some time. Also, the user will return the results of processed tasks to the requester, when they meet in the future.

We say that two mobile users "*encounter*" or "*meet*", which means that they move close and can directly communicate with each other via Bluetooth, or they enter the communication range of some WiFi access points, so that they can indirectly contact each other by using social network softwares via WiFi networks, as shown in Fig. 1. Here, when two users contact each other via WiFi networks, they might visit two different access points at different time, respectively. In this paper, we assume that the communication duration and bandwidth are enough for each user to receive tasks or return results. Additionally, we consider such a mobility model that the inter-meeting time between each user $v_i \in V$ and the requester follows the exponential distribution, whose rate parameter is λ_i. This is because previous studies on mobility models (e.g., [21], [22], [23]) have proved that the inter-meeting time of mobile users in many real MSN traces follows the power law distribution, which can be approximately seen as an exponential distribution. Due to this reason, the mobility model based on the exponential distribution is widely adopted, such as [24], [25]. In this mobility model, the rate parameter λ_i is actually equal to the reciprocal of the expected inter-meeting time between user $v_i \in V$ and the requester, i.e., $\int_0^\infty t\lambda_i e^{-\lambda_i t} dt = \frac{1}{\lambda_i}$. Real

trace analysis has also revealed that the mobile behaviors of users are generally predicable. Therefore, we assume that the requester can record the average value of the historical inter-meeting time between itself and each user, and then it can use the historical value to estimate each rate parameter. In practice, along with the eclipse of the time, the average inter-meeting time might vary gradually, so that the cumulative variation might be non-negligible. For this case, we also assume that the rate parameter is allowed to be updated when the requester meets each user.

2.2 Problem

Consider a mobile crowdsensing in the above MSN. Without loss of generality, we let the requester be user v_0, and suppose that the requester has m indivisible mobile sensing tasks, denoted by $J = \{j_1, j_2, \ldots, j_m\}$. These tasks might be different types of tasks. Despite this, their workloads can be uniformly indicated by the sensing time, denoted as $\tau_1, \tau_2, \ldots, \tau_m$. For simplicity, we assume that all tasks in J need to be assigned to other users, and each task will be assigned to only one user.

In this paper, we focus on the makespan of each task in J, which is defined as follows:

Definition 1. *The* makespan *of a task $j \in J$, denoted by $M(j)$, is the time that the requester finally receives the result of this task, including the time of this task being sent from the requester to some user, the time of this task being conducted by this user, and the time for the result of this task being sent back to the requester.*

Before the problem definition, we define the task assignment solution as follows:

Definition 2. *A* task assignment solution *is a partition of the universal task set J, denoted by Π. More specifically, $\Pi = \{J_1, J_2, \ldots, J_n\}$, in which J_i is the set of tasks that are assigned to user v_i $(1 \le i \le n)$. Moreover, each J_i is an* ordered set *of tasks, satisfying $\sum_{i=1}^{n} J_i = J$ and $J_i \cap J_{i'} = \emptyset$ for $\forall J_i, J_{i'} \in \Pi$. For $\forall j, j' \in J_i$, we use the order $j \preceq j'$ to indicate that task j will be processed* prior *to j' in J_i (if $j \neq j'$).*

In Definition 2, if $J_i = \emptyset$, the requester will not assign any tasks to user v_i. Otherwise, the requester will send the tasks in J_i to user v_i, and then, user v_i will process these tasks according to their orders in J_i.

Based on the above definitions, we define and formalize our problems as follows:

Definition 3. *The* Minimum-Average-Makespan *task assignment problem is to determine a task assignment solution Π for the requester, which can minimize the average makespan of all tasks. Let the average makespan of all tasks in Π be denoted by $AM(\Pi)$. Then, the problem is formalized as follows:*

$$\begin{aligned} Minimize: \quad & AM(\Pi) = \frac{1}{m}\sum_{j \in J} M(j)|_{\Pi} \\ s.t.: \quad & \Pi \text{ is an ordered set partition of } J. \end{aligned} \tag{1}$$

Definition 4. *The* Minimum-Largest-Makespan *task assignment problem is to determine a task assignment solution Π for the requester, which can minimize the largest makespan of all tasks. Let the largest makespan of all tasks in Π be denoted by $LM(\Pi)$. Then, the problem is formalized as follows:*

TABLE 1
Description of Major Notations

Notation	Description
v, V	mobile user and the universal set of users.
λ_i	the rate parameter of exponential distribution of the inter-meeting time between user v_i and the requester.
δ_i	the average time of the requester sending tasks to and receiving results from user v_i (Eq. (4)).
j_k, τ_k, J	the kth task, the workload of task j_k, and the universal set of tasks.
$M(j)$	the makespan of task j (Definition 1).
J_i	the set of tasks assigned to user v_i.
$\Pi,\ \Pi_A,\ \Pi_L$	the task assignment solution (Definition 2), where the subscripts "A" and "L" indicate the solution for the MAM problem and the MLM problem, respectively.
$\Pi_A^*,\ \Pi_L^*$	the task assignment solutions produced by AOTA and LOTA.
$OPT_A,\ OPT_L$	the optimal task assignment solutions of the MAM and MLM problems.
$AM(\Pi)$	the average makespan of tasks for the task assignment solution Π (Definition 3).
$LM(\Pi)$	the largest makespan of tasks for the task assignment solution Π (Definition 4).

$$\begin{aligned} Minimize: \quad & LM(\Pi) = Max\{M(j_1)|_{\Pi}, \ldots, M(j_m)|_{\Pi}\} \\ s.t.: \quad & \Pi \text{ is an ordered set partition of } J. \end{aligned} \tag{2}$$

In the following sections, we use OPT to denote the optimal task assignment solution. To distinguish, we use Π_A, OPT_A and Π_L, OPT_L to denote the task assignment solutions for the MAM and MLM problems, respectively. Additionally, for ease of the following presentation, we list the main notations in Table 1.

3 The MAM Task Assignment

In this section, we propose the online task assignment algorithm, i.e., AOTA, for the MAM task assignment problem. In AOTA, the requester makes multiple rounds of online task assignment decisions, each of which is conducted when it encounters a mobile user, until all tasks are assigned. In each round of online decision, the key problem is how to determine the tasks that the requester should assign to the encountered mobile user. We regard this problem as a virtual offline task assignment among the encountered user and the users whom the requester might encounter in the future, as shown in Fig. 2. Then, we adopt a greedy strategy to conduct the task assignment among these users. However, this is actually just a virtual task assignment. In the results, only the tasks that are assigned to the encountered user are real results (i.e., the final results of online decision), while other tasks (assigned to the users that might be encountered in the future) are virtual assignment results.

In the following, we first derive a formula to compute the average makespan for a given task assignment solution. Then, based on the formula, we present the greedy task assignment strategy for the virtual offline task assignment, as the building block of AOTA. Finally, we propose the AOTA algorithm, followed by the performance analysis.

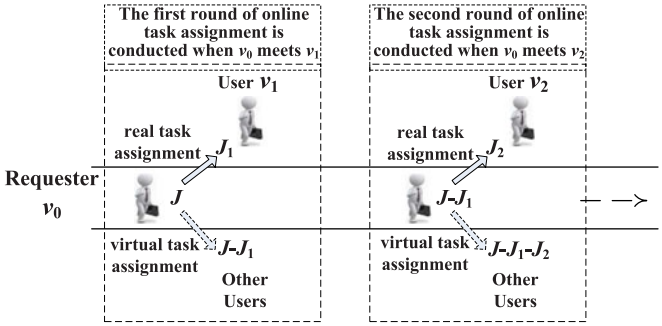


Fig. 2. Illustration of online task assignment: the requester moves along a road, and meets users $v_1, v_2, \ldots$, in turn; when the requester meets a user, it conducts a round of online decision, which can be seen as a virtual offline task assignment (J_1 in the first round and J_2 in the second round are real results, while $J - J_1$ and $J - J_1 - J_2$ are virtual results).

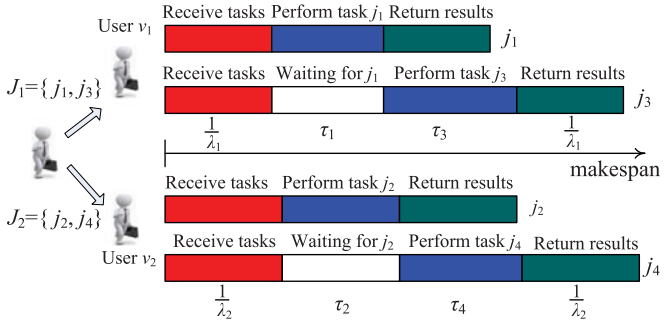


Fig. 3. The average makespan of a task assignment solution $\Pi_A = \{J_1 = \{j_1, j_3\}, J_2 = \{j_2, j_4\}\}$: $AM(\Pi_A) = \frac{1}{4}(M(j_1) + M(j_2) + M(j_3) + M(j_4))$, where $M(j_1) = \frac{2}{\lambda_1} + \tau_1$, $M(j_2) = \frac{2}{\lambda_2} + \tau_2$, $M(j_3) = \frac{2}{\lambda_1} + \tau_1 + \tau_3$, and $M(j_4) = \frac{2}{\lambda_2} + \tau_2 + \tau_4$.

3.1 Basic Formula

Without loss of generality, we consider an arbitrary MAM task assignment solution Π, and derive the average makespan of all tasks for this task assignment solution. More specifically, we have the following theorem:

Theorem 1. *The average makespan $AM(\Pi)$ for the task assignment solution $\Pi = \{J_1, J_2, \ldots, J_n\}$ satisfies*

$$AM(\Pi) = \frac{1}{m}\sum_{i=1}^{n}\sum_{j\in J_i}\left(\delta_i + \sum_{j'\in J_i \wedge j' \preceq j} \tau_{j'}\right), \tag{3}$$

where δ_i is the average time of the requester sending tasks to and receiving results from user v_i, satisfying:

$$\delta_i = \begin{cases} \frac{1}{\lambda_i}, & \text{the requester and user } v_i \text{is encountering;} \\ \frac{2}{\lambda_i}, & \text{otherwise.} \end{cases} \tag{4}$$

Proof. We consider an arbitrary task $j \in J_i$. It involves three phases. At the beginning, the task j will be sent to user v_i by the requester. This happens only when the requester meets user v_i. Hence, if the requester is exactly meeting the user v_i, the time of sending the task will be zero (as the transmission time is too small so that it can be ignored). Otherwise, the time of sending the task is the time that the requester waits for meeting user v_i. Since the inter-meeting time of user v_i and the requester follows the exponential distribution with a rate parameter λ_i, the average time of the requester waiting for user v_i is their expected meeting time, i.e., $\int_0^\infty \lambda_i t e^{-\lambda_i t} dt = \frac{1}{\lambda_i}$. In the second phase, user v_i will perform the sensing tasks in J_i. Note that user v_i will first perform those tasks that are prior to j in J_i. Thus, the time of task j being processed in this phase is $\sum_{j'\in J_i \wedge j' \preceq j} \tau_{j'}$. In the third phase, the result of task j will be returned to the requester by user v_i when they have another meeting. The average time is still equal to the expected meeting time, i.e., $\int_0^\infty \lambda_i t e^{-\lambda_i t} dt = \frac{1}{\lambda_i}$. Thus, the average makespan of this task is $M(j)|_\Pi = \delta_i + \sum_{j'\in J_i \wedge j' \preceq j} \tau_{j'}$, where $\delta_i = \frac{1}{\lambda_i}$, if the requester and user v_i encounter; otherwise, $\delta_i = \frac{2}{\lambda_i}$. By putting $M(j)|_\Pi$ into Eq. (1), we can get that the theorem holds. □

Theorem 1 gives a formula, by which we can compute the average makespan for a given task assignment solution. According to the formula in this theorem, we can derive a basic property of the optimal task assignment solution OPT_A, as follows.

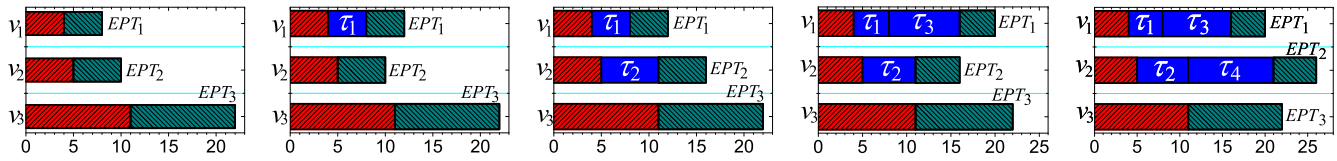
Theorem 2. *Suppose the optimal task assignment solution is $OPT_A = \{J_1, J_2, \ldots, J_n\}$. Then, the tasks with small workloads will be processed first. More specifically, for $\forall j, j' \in J_i$ $(1 \le i \le n)$, if $\tau_j \le \tau_{j'}$, then the order of tasks j and j' in J_i satisfies $j \preceq j'$.*

Proof. The contradiction method is adopted. Assume that there exists the tasks $j, j' \in J_i \in OPT_A$, satisfying $\tau_j \le \tau_{j'}$ but $j' \preceq j$. Without loss of generality, we assume that j' and j are the kth and hth tasks in J_i, respectively, where $k < h$. Then, we construct another task assignment solution Π_A by exchanging the orders of tasks j and j' in J_i. Computing $AM(OPT_A)$ and $AM(\Pi_A)$ according to Eq. (3) in Theorem 1, and comparing them, we have:

$$AM(OPT_A) - AM(\Pi_A) = \frac{1}{m}(h-k)(\tau_{j'} - \tau_j) > 0. \tag{5}$$

This means that the new task assignment solution Π_A can achieve a smaller average makespan value than OPT_A. This is a contradiction to the optimality of OPT_A. Thus, the assumption is incorrect, and we have $j \preceq j'$. □

Theorem 2 shows that the optimal MAM performance can be achieved only when small-workload tasks are processed first. For example, we can compute the average makespan for a task assignment solution in Fig. 3, where $\Pi_A = \{J_1, J_2\}$, $J_1 = \{j_1, j_3\}$, $J_2 = \{j_2, j_4\}$, and $\delta_1 = \frac{2}{\lambda_1}$, $\delta_2 = \frac{2}{\lambda_2}$. The makespan of task j_1 contains the expected time for the requester sending the task to user v_1, the time for v_1 performing the task, and the expected time for v_1 returning the corresponding result. Thus, $M(j_1) = \frac{2}{\lambda_1} + \tau_1$. Besides, the makespan of task j_3 needs to contain the waiting time for v_1 processing task j_1 which is prior to j_3. That is to say, $M(j_3) = \frac{2}{\lambda_1} + \tau_1 + \tau_3$. Likewise, we can get $M(j_2) = \frac{2}{\lambda_2} + \tau_2$ and $M(j_4) = \frac{2}{\lambda_2} + \tau_2 + \tau_4$. Then, the average makespan for the task assignment solution Π_A is $AM(\Pi_A) = \frac{1}{4}(M(j_1) + M(j_2) + M(j_3) + M(j_4))$. In this example, the workloads of tasks j_1, j_2, j_3, and j_4 satisfy $\tau_1 < \tau_2 < \tau_3 < \tau_4$, and Π_A is actually the optimal task assignment solution. In order to achieve the minimum average makespan, users v_1 and v_2 first process j_1 and j_2, respectively.



(a) The initial state (b) Assign the task j_1 (c) Assign the task j_2 (d) Assign the task j_3 (e) Assign the task j_4

Fig. 4. A virtual MAM offline task assignment example: greedily assign tasks j_1, j_2, j_3, j_4 ($\tau_1 = 4, \tau_2 = 6, \tau_3 = 8, \tau_4 = 10$) to users v_1, v_2, v_3 ($\lambda_1 = 1/4, \lambda_2 = 1/5, \lambda_3 = 1/11$), and the optimal MAM offline task assignment solution is $\{J_1, J_2, J_3\}$, where $J_1 = \{j_1, j_3\}$, $J_2 = \{j_2, j_4\}$, and $J_3 = \emptyset$.

3.2 Building Block

Since the whole online MAM task assignment can be seen as a series of virtual MAM offline task assignments, we first present the greedy STFA and EURT strategy for the virtual MAM offline task assignment, as the building block of the online task assignment. Theorem 2 has shown that small-workload tasks need to be assigned first, in order to achieve the optimal MAM performance. Moreover, these tasks need to be assigned to the earliest idle users. To determine the earliest idle users, we define a concept of expected processing time.

Definition 5 (Expected Processing Time). *The expected processing time of a user, denoted by EPT, is the expected time for the user to meet the requester, and perform all tasks in hand, until it returns all results. More specifically, the expected processing time of a user v_i who has the tasks $\{j_{i_1}, j_{i_2}, \ldots, j_{i_k}\}$ is $EPT_i = \delta_i + \tau_{i_1} + \tau_{i_2} + \cdots + \tau_{i_k}$. Specially, when the user has no tasks in hand, its expected processing time is δ_i.*

Without loss of generality, we consider a virtual MAM offline task assignment, in which the unassigned tasks are $\{j_1, j_2, \ldots, j_k\}$ and the unassigned users are $\{v_1, v_2, \ldots, v_i\}$. Moreover, the workloads of these tasks and the initial EPT values of the users satisfy $\tau_1 < \tau_2 < \cdots < \tau_k$ and $\delta_1 \leq \delta_2 \leq \cdots \leq \delta_i$, respectively. The greedy task assignment strategy is that we always select the smallest workload task among the unassigned tasks and assign it to the user with the smallest expected processing time. Concretely, the task assignment is conducted as follows. First, the requester assigns task j_1 to the user with the smallest EPT value. At the beginning, user v_1 has the minimum EPT. Therefore, task j_1 is assigned to user v_1. Second, we focus on task j_2. Now, user v_1 has task j_1 in hand. Its EPT becomes $\delta_1 + \tau_1$. The minimum EPT will be $Min\{\delta_1 + \tau_1, \delta_2, \ldots, \delta_i\}$. Without loss of generality, we assume that user v_2 has the minimum EPT at this time. Then, the requester assigns task j_2 to user v_2. In this way, the remaining tasks are assigned in turn. In Section 3.4, we will show that such a simple greedy strategy can produce an optimal solution for the virtual MAM offline task assignment problem.

Fig. 4 shows the process of greedily determining the optimal MAM offline task assignment solution through a simple example, in which four tasks j_1, j_2, j_3, j_4 are assigned to three users v_1, v_2, v_3. In this example, $\tau_1 = 4, \tau_2 = 6, \tau_3 = 8, \tau_4 = 10$, and $\lambda_1 = \frac{1}{4}, \lambda_2 = \frac{1}{5}, \lambda_3 = \frac{1}{11}$. At the beginning, the EPT values of users v_1, v_2, v_3 are $\frac{2}{\lambda_1}$, $\frac{2}{\lambda_2}$, and $\frac{2}{\lambda_3}$, as shown in Fig. 4a. Then, task j_1 is assigned to user v_1, as shown in Fig. 4b. After this, the EPT value of user v_1 becomes $\frac{2}{\lambda_1} + \tau_1$, which is larger than that of user v_2. Then, task j_2 is assigned to user v_2, as shown in Fig. 4c. In the same way, task j_3 is assigned to user v_1 again in Fig. 4d. Finally, task j_4 is assigned to user v_2 again in Fig. 4e.

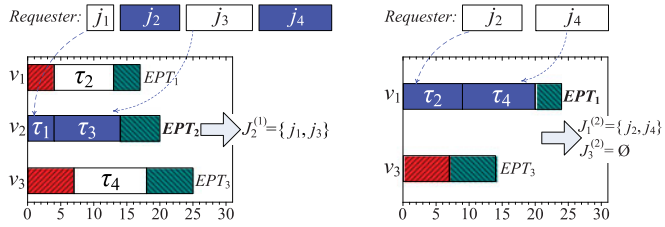
3.3 The AOTA Algorithm

The whole online task assignment in the AOTA algorithm is composed of multiple rounds of online decision, each of which is conducted when the requester encounters a user. Moreover, each round of decision is treated as a virtual MAM offline task assignment problem. We adopt the greedy method in the building block to produce a temporary task assignment result, in which only the tasks that are assigned to the encountered user are real results, while other assigned tasks are virtual results. The virtual results might be assigned in next round of online decision. Denote the task assignment solution produced by AOTA as Π_A^*. Then, the requester conducts the AOTA algorithm as follows.

Algorithm 1. The AOTA Algorithm

Require: $J = \{j_1, j_2, \ldots, j_m : \tau_1 \leq \tau_2 \leq \cdots \leq \tau_m\}$,
$V = \{v_1, v_2, \ldots, v_n : \lambda_1, \lambda_2, \ldots, \lambda_n\}$.
Ensure: Π_A^*
When the requester meets user v_i do
1: **for** each user $v_l \in V$ **do**
2: $J_l = \emptyset$;
3: $EPT_l = \frac{2}{\lambda_l}$;
4: **if** $v_l = v_i$ **then**
5: $EPT_l = \frac{1}{\lambda_l}$;
6: **for** each task $j_k \in J$ from $k = 1$ to $|J|$ **do**
7: $i_{min} = argmin(\{EPT_l \mid v_l \in V\})$;
8: $J_{i_{min}} = J_{i_{min}} + \{j_k\}$;
9: $EPT_{i_{min}} = EPT_{i_{min}} + \tau_k$;
10: Assign the tasks in J_i to user v_i, i.e., $\Pi_A^* = \Pi_A^* + \{J_i\}$;
11: $J = J - \{J_i\}$;
12: $V = V - \{v_i\}$;
13: **return** Π_A^*;

At the beginning, the requester holds all tasks in J. When it encounters a mobile user v_i, it starts the first round of online decision, i.e., the first round of virtual MAM offline task assignment. More specifically, the requester first computes the EPT values of user v_i and the users who have not been met by itself. Then, it adopts the greedy strategy in the building block to assign tasks. That is, it always assigns the task with the smallest workload to the user who has the minimum EPT value, until all tasks are assigned. Through this process, the requester will get a temporary task assignment result $\{J_1^{(1)}, \ldots, J_i^{(1)}, \ldots, J_n^{(1)}\}$ (the superscript (1) indicates the 1th round). However, only $J_i^{(1)}$ among them is the real assignment result, while $J - J_i^{(1)} = \{J_1^{(1)}, \ldots, J_{i-1}^{(1)}, J_{i+1}^{(1)}, \ldots, J_n^{(1)}\}$ are virtual assignment results. The requester only assigns the tasks in $J_i^{(1)}$ to user v_i (i.e., $J_i^{(1)} \in \Pi_A^*$), while keeping the remaining tasks (i.e., the tasks in $J - J_i^{(1)}$) in



(a) the requester determines J_2 when it meets v_2: $J_2^{(1)}=\{j_1,j_3\}$. (b) the requester determines J_1 when it meets v_1: $J_1^{(2)}=\{j_2,j_4\}$.

Fig. 5. An example of online task assignment, in which the requester encounters users v_2, v_1, v_3, in turn.

hand. This ends the first round of task assignment. When the requester encounters another mobile user $v_{i'}$, it starts the second round of virtual MAM offline task assignment. By using the greedy strategy in the building block, the requester virtually assigns the tasks in $J - J_i^{(1)}$ to users $V - \{v_i\}$. This will produce another temporary task assignment result $\{J_1^{(2)}, \ldots, J_{i-1}^{(2)}, J_{i+1}^{(2)}, \ldots, J_{i'}^{(2)}, \ldots, J_n^{(2)}\}$, in which only $J_{i'}^{(2)}$ is the real result, while the others (i.e., $J - J_i^{(1)} - J_{i'}^{(2)}$) are virtual results. Next, the requester might continue to meet other mobile users. For each encounter, it makes a round of task assignment decision in the same way, to assign some tasks to the encountered user, until all tasks are assigned. If the requester meets all users in the order of v_1, $v_2, \ldots, v_n$, the final task assignment solution will be $\Pi_A^* = \{J_1^{(1)}, J_2^{(2)}, \ldots, J_i^{(i)}, \ldots, J_n^{(n)}\}$.

The detailed AOTA algorithm is presented in Algorithm 1. At the beginning, the set of unassigned tasks is J, and the set of users who have not been met by the requester is V. Then, when the requester meets an arbitrary user $v_i \in V$, it first initializes the EPT value and the assigned task set for each user $v_l \in V$ in Steps 1-5: $EPT_l = \frac{2}{\lambda_l}$, $J_l = \emptyset$. Specially, $EPT_i = \frac{1}{\lambda_i}$. From Step 6 to Step 9, the requester determines which tasks in hand should be assigned to the encountered user v_i by using the greedy method in the building block. This will produce a temporal task assignment result, in which J_i is the real result, and the others are virtual results. Hence, the requester assigns the tasks in J_i to the encountered user v_i in Step 10, and updates the unassigned task set as $J = J - J_i$ in Step 11, which means that the requester will hold these unassigned tasks in hand. Meanwhile, the set of users who have not been met by the requester is updated in Step 12. This ends a round of online decision. Next, when the requester meets another user, this algorithm will be conducted again to assign some of the remaining tasks to the new encountered user, and so on, until all tasks are assigned. The computation overhead is dominated by Step 7, which is $O(mn^2)$.

Fig. 5 shows an example, in which the requester has four tasks j_1, j_2, j_3, j_4 ($\tau_1 = 4, \tau_2 = 9, \tau_3 = 10, \tau_4 = 11$), and wishes to assign them to three mobile users v_1, v_2, v_3 ($\lambda_1 = \frac{1}{4}$, $\lambda_2 = \frac{1}{6}, \lambda_3 = \frac{1}{7}$). In Fig. 5, the requester encounters v_2 first. Then, the requester computes the EPT values of v_1, v_2 and v_3: $EPT_1 = \frac{2}{\lambda_1} = 8$, $EPT_2 = \frac{1}{\lambda_2} = 6$, and $EPT_3 = \frac{2}{\lambda_3} = 14$, among which EPT_2 is the smallest. Thus, the requester assigns task j_1, which has the smallest workload, to user v_2. Then, EPT_2 becomes $EPT_2 = \frac{1}{\lambda_2} + \tau_1 = 10$. Next, the requester adopts the same greedy strategy to assign the remaining tasks. As a result, we have $J_1^{(1)} = \{j_2\}$, $J_2^{(1)} = \{j_1, j_3\}$, and $J_3^{(1)} = \{j_4\}$, as shown in Fig. 5a. Among them, only $J_2^{(1)} = \{j_1, j_3\}$ is the real result, and $J_1^{(1)}$, $J_3^{(1)}$ are virtual results. Then, the requester assigns tasks j_1 and j_3 to v_2, while keeping the remaining tasks j_2 and j_4 in hand for future task assignments. In the second round, the requester encounters user v_1. Then, the requester conducts the same task assignment process. The result is $J_1^{(2)} = \{j_2, j_4\}$, and $J_3^{(2)} = \emptyset$, as shown in Fig. 5b. Moreover, $J_1^{(2)} = \{j_2, j_4\}$ is the final result. According to this result, the requester assigns tasks j_2 and j_4 to v_1. Now, there are no remaining tasks. Then, $J_3^{(2)} = \emptyset$ is also the final result. Thus, the final result is $\Pi_A^* = \{J_1^{(2)}, J_2^{(1)}, J_3^{(2)}\}$, where $J_1^{(2)} = \{j_2, j_4\}$, $J_2^{(1)} = \{j_1, j_3\}$, and $J_3^{(2)} = \emptyset$.

3.4 Performance Analysis

In AOTA, each round of online decision is viewed as a virtual MAM offline task assignment, conducted when the requester encounters a mobile user. The greedy STFA and EURT strategy is adopted to produce a temporary result for each round of virtual MAM offline task assignment. Without loss of generality, we assume that the requester meets other users in the order of $v_1, v_2, \ldots, v_n$. Then, the temporary result can be defined as follows:

Definition 6 (Temporary Task Assignment Result). *We use $\Pi_A^{(i)}$ to denote the temporary result for the ith round of virtual MAM offline task assignment (i.e., the case that the requester encounters v_i). More specifically, we let $\Pi_A^{(i)} = \{J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}, J_i^{(i)}, J_{i+1}^{(i)}, \ldots, J_n^{(i)}\}$, in which $J_{i+1}^{(i)}, \ldots, J_n^{(i)}$ are the virtual task assignment results determined by the ith round of decision, and $J_1^{(1)}, \ldots, J_i^{(i)}$ are the real task assignment results determined by the 1st, ..., ith rounds of decisions, respectively. Moreover, we use $OPT_A^{(i)}$ to denote the optimal temporary result for the ith round of virtual MAM offline task assignment.*

To analyze the competitive ratio of AOTA, we first prove that the greedy STFA and EURT strategy can achieve the optimal MAM performance for each virtual offline task assignment. Theorem 2 has shown the optimality of the greedy STFA strategy. Here, we have the following theorem to demonstrate the optimality of the greedy EURT strategy for the offline task assignment:

Theorem 3. *Consider an arbitrary ith round of virtual MAM offline task assignment. Without loss of generality, suppose that tasks $j_1, j_2, \ldots, j_{m'}$ ($1 \le m' \le m$) need to be assigned to $v_i, \ldots, v_n$ in this round, while other tasks have been assigned to $v_1, \ldots, v_{i-1}$ before this round. Moreover, the workloads of these tasks are assumed to satisfy $\tau_1 \le \tau_2 \le \cdots \le \tau_{m'}$. Among the m' tasks, we assume that tasks $j_1, j_2, \ldots, j_{k-1}$ ($1 \le k \le m'$) have been assigned to $v_i, \ldots, v_n$. Let the set of tasks that have been assigned to user v_l ($i \le l \le n$) be J_l and the current expected processing time of this user be EPT_l. Then, $J_i^{(i)}$, $J_{i+1}^{(i)}, \ldots, J_n^{(i)}$ in the optimal virtual MAM offline task assignment solution $OPT_A^{(i)}$ satisfies:*

1) *The user who currently has the minimum expected processing time will be assigned the maximum number of tasks in the following rounds of task assignments:*

$$EPT_l = Min\{EPT_i, \ldots, EPT_n\} \Rightarrow \\ |J_l^{(i)} - J_l| = Max\{|J_i^{(i)} - J_i|, \ldots, |J_n^{(i)} - J_n|\}, \tag{6}$$

where $|J_l^{(i)} - J_l|$ is the number of tasks in set $J_l^{(i)} - J_l$.

2) *The task j_k will be assigned to the user who currently has the minimum expected processing time:*

$$EPT_l = Min\{EPT_i, \ldots, EPT_n\} \Rightarrow j_k \in J_l^{(i)}. \tag{7}$$

Proof. 1) First, we prove part 1 by using the contradiction. Assume that although $EPT_l = Min\{EPT_1, \ldots, EPT_n\}$, $J_l^{(i)} - J_l$ is not the set with the maximum tasks among $\{J_i^{(i)} - J_i, \ldots, J_n^{(i)} - J_n\}$. Without loss of generality, we let $|J_{l'}^{(i)} - J_{l'}| = Max\{|J_i^{(i)} - J_i|, \ldots, |J_n^{(i)} - J_n|\}$ $(i \le l' \le n, l' \neq l)$. Moreover, we assume that $J_l^{(i)} - J_l = \{j_{l_1}, \ldots, j_{l_s}\}$, $J_{l'}^{(i)} - J_{l'} = \{j_{l'_1}, \ldots, j_{l'_r}\}$, and $r > s$. Then, we construct another task assignment solution $\Pi_A = \{J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}, \ldots, \bar{J}_l^{(i)}, \ldots, \bar{J}_{l'}^{(i)}, \ldots, J_n^{(i)}\}$, where $\bar{J}_l^{(i)} - J_l = \{j_{l'_1}, \ldots, j_{l'_r}\}$, $\bar{J}_{l'}^{(i)} - J_{l'} = \{j_{l_1}, \ldots, j_{l_s}\}$, by exchanging the tasks in $J_l^{(i)} - J_l$ and $J_{l'}^{(i)} - J_{l'}$. After a careful computation on $AM(OPT_A^{(i)})$ and $AM(\Pi_A)$ according to Theorem 2 and Eq. (3) in Theorem 1, we have:

$$AM(OPT_A^{(i)}) - AM(\Pi_A) = \frac{1}{m}(s-r)(EPT_l - EPT_{l'}) > 0. \tag{8}$$

Thus, the new task assignment solution Π_A can achieve a smaller average makespan than $OPT_A^{(i)}$. This is a contradiction to the optimality of $OPT_A^{(i)}$. Due to the contradiction, we have that part 1 of the theorem holds.

2) We still adopt the contradiction method for part 2. Assume that $EPT_l = Min\{EPT_i, \ldots, EPT_n\}$, but $j_k \notin J_l^{(i)}$ (i.e., $j_k \notin J_l^{(i)} - J_l$). Without loss of generality, we assume that task j_k is assigned to user $v_{l'}$ in the optimal offline task assignment solution $OPT_A^{(i)}$, i.e., $j_k \in J_{l'}^{(i)} - J_{l'}$ $(l' \neq l)$, and assume that the task with the smallest workload in the task set $J_l^{(i)} - J_l$ is j_h $(k < h \le m')$. According to Theorem 2, tasks j_k and j_h are the first tasks to be processed in $J_{l'}^{(i)} - J_{l'}$ and $J_l^{(i)} - J_l$, respectively. Now, we construct another task assignment solution Π_A by exchanging the tasks $j_k \in J_{l'}^{(i)} - J_{l'}$ and $j_h \in J_l^{(i)} - J_l$. Then, computing $AM(OPT_A^{(i)})$ and $AM(\Pi_A)$ according to Eq. (3) in Theorem 1, and comparing them, we have:

$$AM(OPT_A^{(i)}) - AM(\Pi_A) = \frac{1}{m}(|J_{l'}^{(i)} - J_{l'}| - |J_l^{(i)} - J_l|)(\tau_k - \tau_h). \tag{9}$$

Since $EPT_l = Min\{EPT_i, \ldots, EPT_n\}$, we have $|J_l^{(i)} - J_l| > |J_{l'}^{(i)} - J_{l'}|$ according to the proof of part 1. Moreover, we have $\tau_k < \tau_h$ due to $k < h$. Thus, we can get

$$AM(OPT_A^{(i)}) - AM(\Pi_A) > 0. \tag{10}$$

This shows that the new task assignment solution Π_A can achieve a smaller average makespan than $OPT_A^{(i)}$. This is a contradiction to the optimality of $OPT_A^{(i)}$. Thus, the assumption is incorrect, and we have $j_k \in J_l^{(i)} - J_l \subseteq J_l^{(i)}$. The theorem holds. □

Theorems 2 and 3 show the optimality of the greedy STFA and EURT strategy. According to this strategy, the task with the smallest workload should always be assigned to the user who has the minimum expected processing time in each round of virtual offline task assignment. This is exactly the strategy adopted by the AOTA algorithm. Thus, we directly have:

Corollary 4. *The AOTA algorithm can achieve the optimal MAM performance for each round of virtual offline task assignment, i.e., $OPT_A^{(i)} = \Pi_A^{(i)}$ for $\forall i \in [1, n]$.*

Here, it is important to highlight that the temporary task assignment result produced in each round is optimal only for that round of decision. From the view of the whole online task assignment process, it is not globally optimal. When the requester encounters another user after this round, the virtually assigned tasks might be re-assigned. Accordingly, the MAM performance will be improved. Along with the more users the requester encounters, the MAM performance will become better and better. We show this in the following theorem:

Theorem 5. *Assume that the requester meets other users in the order of $v_1, v_2, \ldots, v_n$. Then, the temporary task assignment results satisfy $AM(\Pi_A^{(0)}) \ge AM(\Pi_A^{(1)}) \ge \cdots \ge AM(\Pi_A^{(n)})$. Here, the 0th round refers to the virtual MAM offline task assignment before the requester encounters any other users.*

Proof. We consider the ith round of virtual MAM offline task assignment result $\Pi_A^{(i)} = \{J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}, J_i^{(i)}, J_{i+1}^{(i)}, \ldots, J_n^{(i)}\}$ and the $(i-1)$th round of virtual offline task assignment result $\Pi_A^{(i-1)} = \{J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}, J_i^{(i-1)}, J_{i+1}^{(i-1)}, \ldots, J_n^{(i-1)}\}$ $(1 \le i \le n)$. Since $J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}$ are the real task assignment results which are determined before the ith round, they are the same in $\Pi_A^{(i)}$ and $\Pi_A^{(i-1)}$. Moreover, $J_i^{(i-1)}, J_{i+1}^{(i-1)}, \ldots, J_n^{(i-1)}$ are the virtual task assignment results in the $(i-1)$th round, which are re-assigned as $J_i^{(i)}, J_{i+1}^{(i)}, \ldots, J_n^{(i)}$ in the ith round when the requester encounters user v_i. According to Corollary 4, the re-assignment can make the requester achieve the optimal MAM performance in the ith round. This means $AM(\Pi_A^{(i-1)}) \ge AM(\Pi_A^{(i)})$. Due to the arbitrariness of i, we can get the correctness of this theorem. □

Based on the above analysis, we give the competitive ratio of the AOTA algorithm now, which is ratio of the average makespan values of AOTA and the globally optimal online task assignment solution.

Theorem 6. *Assume that there is a god, who can foresee the mobilities of all mobile users, so that it knows at what time the*

requester will meet which user. Based on this, the god can give a globally optimal online task assignment solution, denoted by OPT_A. Then, we have:

1) *The average makespan for the task assignment solution Π_A^* produced by AOTA satisfies:*

$$AM(\Pi_A^*) - AM(OPT_A) \leq \sum_{i=1}^{n} \frac{2}{\lambda_i}. \tag{11}$$

2) *The competitive ratio of the AOTA algorithm satisfies:*

$$\frac{AM(\Pi_A^*)}{AM(OPT_A)} \leq 1 + \frac{n\sum_{i=1}^{n}\frac{2}{\lambda_i}}{\sum_{j=1}^{m}\tau_j}. \tag{12}$$

Proof. Without loss of generality, we assume that the requester meets users $v_1, v_2, \ldots, v_n$ at the time $t_1, t_2, \ldots, t_n$, and it also takes the time $t_1', t_2', \ldots, t_n'$ for these users to return their results to the requester, respectively. Moreover, we assume that the globally optimal solution given by the god is $OPT_A = \{J_1^*, \ldots, J_n^*\}$. Since the god has known the time at which the requester meets other users and the time of other users returning results, this globally optimal task assignment can be seen as a special offline task assignment. Then, we can use Eq. (3) in Theorem 1 to calculate the average makespan:

$$AM(OPT_A) = \frac{1}{m}\sum_{i=1}^{n}\left((t_i + t_i')|J_i^*| + \sum_{j\in J_i^*}\sum_{j'\in J_i^*\wedge j'\preceq j}\tau_{j'}\right). \tag{13}$$

Furthermore, we consider another special offline task assignment case, where there is no god, and the requester has not encountered any other users. Although there is no god, the requester in this case still uses OPT_A as its offline task assignment solution, denoted by Π_A. Then, according to Theorem 1, we have:

$$\begin{aligned} AM(\Pi_A) &= \frac{1}{m}\sum_{i=1}^{n}\left(\frac{2}{\lambda_i}|J_i^*| + \sum_{j\in J_i^*}\sum_{j'\in J_i^*\wedge j'\preceq j}\tau_{j'}\right) \\ &= AM(OPT_A) + \frac{1}{m}\sum_{i=1}^{n}\left(\frac{2}{\lambda_i} - t_i - t_i'\right)|J_i^*|. \end{aligned} \tag{14}$$

Note that, as an offline task assignment solution without a god, Π_A is not optimal. Actually, we have $AM(\Pi_A) \geq AM(\Pi_A^{(0)})$ according to Corollary 4. On the other hand, according to Theorem 5, we have $AM(\Pi_A^{(0)}) \geq AM(\Pi_A^{(n)}) = AM(\Pi_A^*)$. Thus, we can get $AM(\Pi_A) \geq AM(\Pi_A^*)$. Replacing $AM(\Pi_A)$ in this inequality by using Eq. (14), we have:

$$\begin{aligned} AM(\Pi_A^*) - AM(OPT_A) &\leq \frac{1}{m}\sum_{i=1}^{n}\left(\frac{2}{\lambda_i} - t_i - t_i'\right)|J_i^*| \\ &\leq \frac{1}{m}\sum_{i=1}^{n}\frac{2m}{\lambda_i} = \sum_{i=1}^{n}\frac{2}{\lambda_i}. \end{aligned} \tag{15}$$

Thus, the part 1 of this theorem is correct. Further, we can straightforwardly get $AM(OPT_A) \geq \frac{1}{n}\sum_{j=1}^{m}\tau_j$. Therefore, according to Eq. (15), we have:

$$\frac{AM(\Pi_A^*)}{AM(OPT_A)} \leq 1 + \frac{\sum_{i=1}^{n}\frac{2}{\lambda_i}}{AM(OPT_A)} \leq 1 + \frac{n\sum_{i=1}^{n}\frac{2}{\lambda_i}}{\sum_{j=1}^{m}\tau_j}. \tag{16}$$

Thus, this theorem holds. □

Theorem 6 shows that the absolute error of AOTA is no more than a fixed value, i.e., $\sum_{i=1}^{n}\frac{2}{\lambda_i}$, which only depends on the expected meeting time between the requester and other users. When the average expected meeting time is very small, our algorithm can even achieve the nearly optimal result. On the other hand, the competitive ratio of AOTA is subject to both the average workload of tasks and the expected meeting time. When the average workload is very large, the competitive ratio of AOTA will be very close to 1, although the absolute error might change not much. In fact, our simulation results in Section 5 have also captured these observations.

4 The MLM Task Assignment

In this section, we propose the Largest makespan sensitive Online Task Assignment (LOTA) algorithm. In LOTA, the requester also makes multiple rounds of online task assignment decisions, each of which is seen as a virtual MLM offline task assignment and is conducted when the requester encounters a mobile user, until all tasks are assigned. Nevertheless, the virtual MLM offline task assignment is an NP-hard problem. The greedy LTFA and EURT task assignment strategy lets the small-workload tasks be conducted latest, so that it can make the latest task be completed as soon as possible. Therefore, LOTA adopts this greedy strategy in each round of virtual MLM offline task assignment, so as to obtain a nearly optimal solution.

In the following, we first derive a formula to compute the largest makespan for a given task assignment solution. Then, we present the task assignment strategy for the virtual offline task assignment, as the building block of LOTA. Finally, we propose the LOTA algorithm, followed by the performance analysis.

4.1 Basic Formula

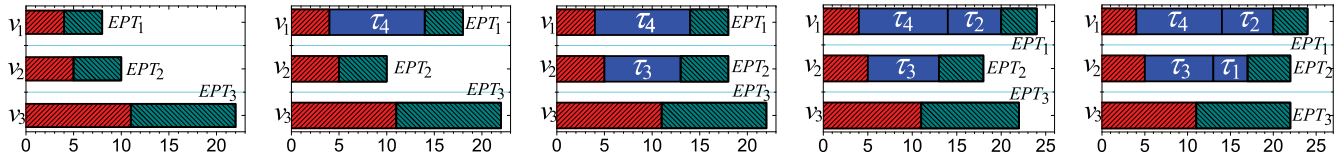
Without lose of generality, we consider an arbitrary task assignment solution $\Pi_L = \{J_1, \ldots, J_n\}$ for the MLM task assignment problem, and compute the largest makespan of all tasks in the solution Π_L. Since the largest makespan of all tasks is the makespan of the task to be completed latest, it is actually equal to the largest expected processing time of all users, according to Definition 4. That is, we have:

$$\begin{aligned} LM(\Pi_L) &= Max\{EPT_1, \ldots, EPT_n\} \\ &= Max\left\{\delta_1 + \sum_{j_1\in J_1}\tau_{j_1}, \ldots, \delta_n + \sum_{j_n\in J_n}\tau_{j_n}\right\}. \end{aligned} \tag{17}$$

Eq. (17) shows that the largest makespan $LM(\Pi_L)$ only depends on the latest performed task. The performing order of the tasks in each J_i will not affect the largest makespan value, different from the average makespan.

4.2 Building Block

Likewise, each round of online decision in LOTA is viewed as a virtual MLM offline task assignment. However, different from AOTA, determining the MLM task assignment solution



(a) The initial state (b) Assign the task j_4 (c) Assign the task j_3 (d) Assign the task j_2 (e) Assign the task j_1

Fig. 6. A virtual MLM offline task assignment example: greedily assign tasks j_4, j_3, j_2, j_1 ($\tau_4 = 10, \tau_3 = 8, \tau_2 = 6, \tau_1 = 4$) to users v_1, v_2, v_3 ($\lambda_1 = 1/4, \lambda_2 = 1/5, \lambda_3 = 1/11$), and the corresponding task assignment solution is $\{J_1, J_2, J_3\}$, where $J_1 = \{j_2, j_4\}$, $J_2 = \{j_1, j_3\}$, and $J_3 = \emptyset$.

for each round of virtual offline task assignment in LOTA is NP-hard. This can be proved by the following theorem:

Theorem 7. *The (virtual) MLM offline task assignment problem is NP-hard.*

Proof. According to Eq. (17), the MLM offline task assignment problem can be seen as an MLM parallel machines scheduling problem. Actually, the n mobile users can be seen as n identical parallel machines, and $\delta_1, \ldots, \delta_n$ can be seen as the start time of the n parallel machines. Then, the problem becomes the MLM job scheduling problem for n identical parallel machines with different start time, which is a well-known NP-hard problem. Therefore, the MLM offline task assignment problem is NP-hard. □

For the virtual MLM offline task assignment in each round of decision, we adopt the greedy LTFA and EURT strategy to assign tasks. Consider an arbitrary round of virtual MLM offline task assignment, in which the remaining unassigned tasks are assumed to be $\{j_1, j_2, \ldots, j_k\} \subseteq J$, and their workloads satisfy $\tau_1 \geq \tau_2 \geq \cdots \geq \tau_k$. The detailed task assignment is conducted as follows. First, the requester computes the EPT values for the users who have not been met. Without loss of generality, they satisfy $\delta_1 \leq \delta_2 \leq \delta_3 \leq \cdots$. Since user v_1 has the smallest EPT value, task j_1 is virtually assigned to this user. Next, the requester updates the EPT value of user v_1 by using $\delta_1 + \tau_1$, and determines the new smallest EPT value, i.e., $Min\{\delta_1 + \tau_1, \delta_2, \delta_3, \cdots\}$. We assume that user v_2 has the minimum EPT at this time. Then, the requester assigns task j_2 to user v_2. In this way, all of the remaining tasks are virtually assigned, in turn.

Fig. 6 illustrates a simple example, in which four tasks j_1, j_2, j_3, j_4 are assigned to three users v_1, v_2, v_3. In this example, $\tau_4 = 10, \tau_3 = 8, \tau_2 = 6, \tau_1 = 4$, and $\lambda_1 = \frac{1}{4}, \lambda_2 = \frac{1}{5}, \lambda_3 = \frac{1}{11}$. At the beginning, the EPT values of users v_1, v_2, v_3 are $\frac{2}{\lambda_1}$, $\frac{2}{\lambda_2}$, and $\frac{2}{\lambda_3}$, as shown in Fig. 6a. Then, task j_4 is assigned to user v_1, as shown in Fig. 6b. After this, the EPT value of user v_1 becomes $\frac{2}{\lambda_1} + \tau_4$, which is larger than that of user v_2. Then, task j_3 is assigned to user v_2, as shown in Fig. 6c. In the same way, task j_2 is assigned to user v_1 again in Fig. 6d. Finally, task j_1 is assigned to user v_2 again in Fig. 6e.

4.3 The LOTA Algorithm

The LOTA algorithm is composed of multiple rounds of virtual MLM offline task assignments. Each round of virtual MLM offline task assignment is conducted when the requester encounters a user. Moreover, the task assignment result is produced by using the greedy LTFA and EURT strategy in the building block. In this result, only the tasks that are assigned to the encountered user are real results, while other assigned tasks are virtual results, which might be re-assigned in next round.

Algorithm 2. The LOTA Algorithm

Require: $J = \{j_1, j_2, \ldots, j_m : \tau_1 \geq \tau_2 \geq \cdots \geq \tau_m\}$,
$V = \{v_1, v_2, \ldots, v_n : \lambda_1, \lambda_2, \ldots, \lambda_n\}$.
Ensure: Π_L^*
When the requester meets user v_i do
1: **for** each user $v_l \in V$ **do**
2: $J_l = \emptyset$;
3: $EPT_l = \frac{2}{\lambda_l}$;
4: **if** $v_l = v_i$ **then**
5: $EPT_l = \frac{1}{\lambda_l}$;
6: **for** each task $j_k \in J$ from $k = 1$ to $|J|$ **do**
7: $i_{min} = argmin(\{EPT_l \mid v_l \in V\})$;
8: $J_{i_{min}} = J_{i_{min}} + \{j_k\}$;
9: $EPT_{i_{min}} = EPT_{i_{min}} + \tau_k$;
10: Assign the tasks in J_i to user v_i, i.e., $\Pi_L^* = \Pi_L^* + \{J_i\}$;
11: $J = J - \{J_i\}$;
12: $V = V - \{v_i\}$;
13: **return** Π_L^*;

The detailed LOTA algorithm is presented in Algorithm 2. At the beginning, the set of unassigned tasks is J, and the set of users who have not been met by the requester is V. Then, when the requester meets an arbitrary user $v_i \in V$, it conducts a round of virtual MLM offline task assignment. First, the requester initializes the EPT value and the assigned task set for each user $v_l \in V$ (Steps 1-5). Next, it determines which tasks in hand should be assigned to the encountered user v_i by using the greedy LTFA and EURT strategy (Steps 6-9). This will produce a temporary offline task assignment result, in which J_i is the real result, and $\{J_1, \ldots, J_{i-1}, J_{i+1}, \ldots, J_n\}$ are virtual results. Then, the requester assigns the tasks in J_i to the encountered user v_i, and updates the unassigned task set as $J = J - J_i$ (Steps 10-11). Meanwhile, the set of users who have not been met by the requester is also updated (Step 12). This ends a round of online task assignment. When the requester meets another user, this algorithm will be conducted again to assign some of the unassigned tasks to the new encountered user, until all tasks are assigned. The computation overhead is dominated by Step 7, which is $O(mn^2)$.

4.4 Performance Analysis

Without loss of generality, we assume that the requester meets other users in the order of $v_1, v_2, \ldots, v_n$. Moreover, we use $\Pi_L^{(i)} = \{J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}, J_i^{(i)}, J_{i+1}^{(i)}, \ldots, J_n^{(i)}\}$ to denote the temporary result for the ith round of virtual MLM offline task assignment, and let $OPT_L^{(i)}$ denote the optimal temporary solution for this round. To analyze the competitive

ratio of LOTA, we first give an error estimation for each virtual MLM offline task assignment solution produced by using the LTFA and EURT strategy.

Theorem 8. *Let $\tau_{max} = Max\{\tau_1, \tau_2, \ldots, \tau_m\}$. Then, for the arbitrary ith ($0 \le i \le n$) round of virtual MLM offline task assignment (here, the 0th round refers to the virtual MLM offline task assignment before the requester encounters any other users), we have:*

$$LM(OPT_L^{(i)}) \le LM(\Pi_L^{(i)}) < LM(OPT_L^{(i)}) + \tau_{max}. \quad (18)$$

Proof. First, we consider the optimal MLM offline task assignment solution $OPT_L^{(i)}$. Obviously, the best $OPT_L^{(i)}$ can be achieved, only when all users complete their received tasks at the same time. That is,

$$LM(OPT_L^{(i)}) \ge \frac{1}{n}\left(\sum_{h=1}^{n}\delta_h + \sum_{j=1}^{m}\tau_j\right). \quad (19)$$

Now, we focus on the task assignment solution $\Pi_L^{(i)}$. Without loss of generality, we assume that task j_k is the last one to be completed, and this task is assigned to user v_l. Then, we have:

$$LM(\Pi_L^{(i)}) = EPT_l \le \frac{1}{n}\sum_{h=1}^{n} EPT_h + \tau_k \quad (20)$$

$$= \frac{1}{n}\left(\sum_{h=1}^{n}\delta_h + \sum_{j=1}^{m}\tau_j\right) + \tau_k \quad (21)$$

$$\le LM(OPT_L^{(i)}) + \tau_{max}. \quad (22)$$

On the other hand, $LM(OPT_L^{(i)}) \le LM(\Pi_L^{(i)})$ due to the optimality of $OPT_L^{(i)}$. Thus, the theorem holds. □

Second, like AOTA, we can also derive that along with the increasing of the rounds of online decision, the MLM performance of the temporarily optimal virtual task assignment solutions will become better and better, as shown in the following theorem:

Theorem 9. *Suppose that the requester meets other users in the order of $v_1, v_2, \ldots, v_n$. Then, the temporarily optimal MLM task assignment solutions satisfy $LM(OPT_L^{(0)}) \ge LM(OPT_L^{(1)}) \ge \cdots \ge LM(OPT_L^{(n)})$.*

Proof. We consider the ith round of temporarily optimal MLM offline task assignment solution $OPT_L^{(i)} = \{J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}, J_i^{(i)}, J_{i+1}^{(i)}, \ldots, J_n^{(i)}\}$ and the $(i-1)$th round of temporarily optimal solution $OPT_L^{(i-1)} = \{J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}, J_i^{(i-1)}, J_{i+1}^{(i-1)}, \ldots, J_n^{(i-1)}\}$ ($1 \le i \le n$). Since $J_1^{(1)}, \ldots, J_{i-1}^{(i-1)}$ are the real task assignment results which are determined before the ith round, they are the same in $OPT_L^{(i)}$ and $OPT_L^{(i-1)}$. Moreover, $J_i^{(i-1)}, J_{i+1}^{(i-1)}, \ldots, J_n^{(i-1)}$ are the virtual task assignment results in the $(i-1)$th round, which are re-assigned as $J_i^{(i)}, J_{i+1}^{(i)}, \ldots, J_n^{(i)}$ in the ith round when the requester encounters user v_i. According to the optimality of $OPT_L^{(i)}$, the re-assignment can make the requester achieve the optimal MLM performance in the ith round. This means $LM(OPT_L^{(i-1)}) \ge LM(OPT_L^{(i)})$. Due to the arbitrariness of i, we can get the correctness of this theorem. □

Based on the above analysis, we can give the competitive ratio of the LOTA algorithm now, which is ratio of the largest makespan of AOTA and the globally optimal online task assignment solution OPT_L.

Theorem 10. *Assume that there is a god, who can foresee the mobilities of all mobile users, so that it knows at what time the requester will meet which user. Based on this, the god can give a globally optimal online task assignment solution OPT_L. Then, we have:*

1) *The largest makespan of the task assignment solution Π_L^* produced by LOTA satisfies:*

$$LM(\Pi_L^*) - LM(OPT_L) \le \tau_{max} + \frac{2}{\lambda_{min}}, \quad (23)$$

where $\tau_{max} = Max\{\tau_1, \tau_2, \ldots, \tau_m\}$, and $\lambda_{min} = Min\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$.

2) *The competitive ratio of the LOTA algorithm satisfies:*

$$\frac{LM(\Pi_L^*)}{LM(OPT_L)} \le 2 + \frac{2}{\lambda_{min}\tau_{max}}. \quad (24)$$

Proof. Without loss of generality, we assume that the requester meets users $v_1, v_2, \ldots, v_n$ at the time $t_1, t_2, \ldots, t_n$, and it also takes the time $t_1', t_2', \ldots, t_n'$ for these users to return their results to the requester, respectively. Moreover, we assume that the globally optimal solution given by the god is $OPT_L = \{J_1^*, \ldots, J_n^*\}$. Since the god has known the time at which the requester meets other users, and the time of these users returning results, this globally optimal task assignment can be seen as a special offline task assignment. Then, we can use Eq. (17) to calculate the largest makespan:

$$LM(OPT_L) = Max\left\{t_1 + t_1' + \sum_{j_1 \in J_1*}\tau_{j_1}, \ldots, t_n + t_n' + \sum_{j_n \in J_n*}\tau_{j_n}\right\}. \quad (25)$$

Furthermore, we consider another special offline task assignment case, where there is no god, and the requester has not encountered any other users. Although there is no god, the requester in this case still uses OPT_A as its offline task assignment solution, denoted by Π_L. Then, according to Eq. (17), we have:

$$\begin{aligned} LM(\Pi_L) &= Max\left\{\frac{2}{\lambda_1} + \sum_{j_1 \in J_1*}\tau_{j_1}, \ldots, \frac{2}{\lambda_n} + \sum_{j_n \in J_n*}\tau_{j_n}\right\} \\ &\le LM(OPT_L) + Max\left\{\frac{2}{\lambda_1} - t_1 - t_1', \ldots, \frac{2}{\lambda_n} - t_n - t_n'\right\} \\ &\le LM(OPT_L) + Max\left\{\frac{2}{\lambda_1}, \ldots, \frac{2}{\lambda_n}\right\}. \end{aligned} \quad (26)$$

Note that, as an offline task assignment solution without a god, Π_L is not optimal. Hence, we have $LM(\Pi_L) \ge LM(OPT_L^{(0)})$. On the other hand, according to Theorem 9,

we have $LM(OPT_L^{(0)}) \geq LM(OPT_L^{(n)})$. Thus, we can get $LM(\Pi_L) \geq LM(OPT_L^{(n)})$. Furthermore, according to Theorem 8, we have:

$$\begin{aligned} LM(\Pi_L^*) = LM(\Pi_L^{(n)}) &\leq LM(OPT_L^{(n)}) + \tau_{max} \\ &\leq LM(\Pi_L) + \tau_{max}. \end{aligned} \tag{27}$$

Replacing $LM(\Pi_L)$ in Eq. (27) by using Eq. (26), we have:

$$\begin{aligned} LM(\Pi_L^*) - LM(OPT_L) &\leq \tau_{max} + Max\left\{\frac{2}{\lambda_1}, \ldots, \frac{2}{\lambda_n}\right\} \\ &= \tau_{max} + \frac{2}{\lambda_{min}}. \end{aligned} \tag{28}$$

Thus, the part 1 of this theorem is correct. Straightforwardly, we have $LM(OPT_L) \geq \tau_{max}$. Therefore, according to Eq. (28), we have:

$$\frac{LM(\Pi_L^*)}{LM(OPT_L)} \leq 1 + \frac{\tau_{max} + \frac{2}{\lambda_{min}}}{LM(OPT_L)} \leq 2 + \frac{2}{\lambda_{min}\tau_{max}}. \tag{29}$$

Thus, this theorem holds. □

Theorem 10 shows that the absolute error of LOTA is no more than a fixed value, i.e., $\tau_{max} + \frac{2}{\lambda_{min}}$, dominated by the largest workload and the average expected meeting time between the requester and other users. When the average expected meeting time and the workloads of tasks are very small, our algorithm can even achieve the nearly optimal result.

5 Evaluation

We conduct extensive simulations to evaluate the performances of the proposed algorithms. The compared algorithms, the traces that we used, the simulation settings, and the results are presented as follows.

5.1 Algorithms in Comparison

To the best of our knowledge, AOTA and LOTA are the first online task assignment algorithms for crowdsensing in mobile social networks that take the time of delivering tasks and results into consideration. Existing crowdsensing task assignment algorithms cannot be tailored to address our problems. Consequently, we select three most related algorithms from MSNs and the parallel machine scheduling area for comparison:

First, we implement the Water Filling (*WF*) algorithm which is a task allocation algorithm in MSNs [26]. The WF algorithm assigns tasks, in turn, to the earliest idle user. Different from our algorithms, the tasks in WF are assigned according to their initial orders. Since the initial orders of tasks are generally random, WF can also be viewed as assigning tasks in a random order. Second, we realize the classic task schedule algorithm in the parallel machine scheduling area, denoted by *LF* (Largest-task-First-process) [27]. The LF algorithm assigns tasks to earliest idle user/machine in the decreasing order of their workloads, which actually is the same as the task assignment strategy for the virtual offline task assignment in LOTA. Third, for the integrity, we also implement the *SF* (Smallest-task-First-process) algorithm, in which tasks are assigned to earliest idle user in the ascending order of their workloads. This actually is the same as the task assignment strategy for the virtual offline task assignment in AOTA. Note that, the three algorithms exactly consist of the most typical task assignment strategies.

In addition, as the benchmark of the evaluation for the MAM performance, we design the optimal MAM online task assignment algorithm OPT_A according to Theorem 6. Here, we have not implemented the OPT_L algorithm due to the NP-hardness of the MLM problem. Moreover, we consider multiple requesters in the simulations. It is possible that multiple requesters assign tasks to a same user. In this case, the user only receives the tasks from the first requester, until it has finished these tasks. The remaining requesters will re-assign their tasks.

TABLE 2
Statistics of the Real Traces

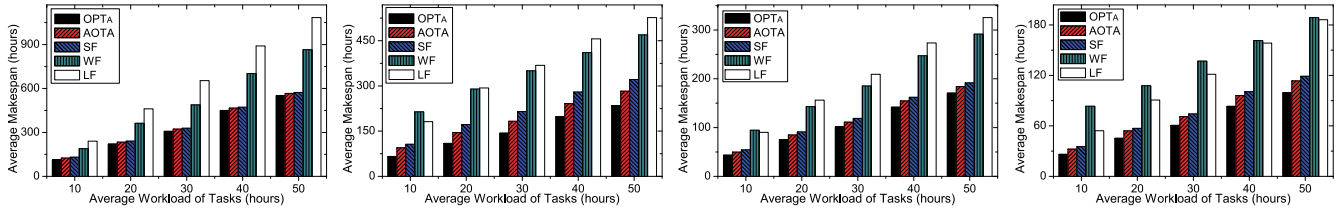
Trace	Contacts	Length (hours)	Requester	Other users
Intel	2,766	99.8	9	128
Cambridge	6,732	145.6	12	223
Infocom	28,216	76.6	41	264
UMassDieselNet	227,657	95.3	4	36

5.2 Real-Traces Used and Simulation Settings

The *Cambridge Haggle Trace* [28] includes three traces of Bluetooth device connections by people carrying mobile devices (iMotes) over a certain number of days. These traces are collected by different groups of people in office environments, conference environments, and city environments, respectively. The nodes in the trace are classified into two groups: internal nodes and external nodes. Since there is no meeting record between internal nodes, we only use these internal nodes as requesters, and let the external nodes receive and process tasks. Table 2 shows some statistics of the traces that we used.

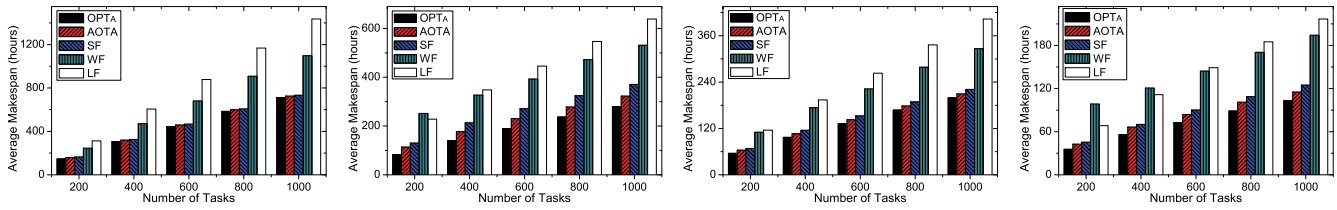
The *UMassDieselNet Trace* [29] contains the bus-to-bus contacts (the durations of which are relatively short) of 40 buses. Our simulations are performed on traces collected over 55 days during the Spring 2006 semester, with weekends, Spring break, and holidays removed due to reduced schedules. The bus system serves approximately ten routes. There are multiple shifts serving each of these routes. Shifts are further divided into morning (AM), midday (MID), afternoon (PM), and evening (EVE) sub-shifts. Drivers choose buses at random to run the AM sub-shifts. At the end of the AM sub-shift, the bus is often handed over to another driver to operate the next sub-shift on the same route, or on another route. For this trace, we select 4 buses with long trace records as the requesters, and let the remaining buses act as other mobile users.

In addition, we estimate the rate parameter λ_i of each user v_i by using the ratio of its meeting times with requesters and the total duration in the traces. Moreover, we randomly produce the tasks for each requester. The number of tasks is selected from $\{200,400,\ldots,1{,}000\}$. The average workload of all tasks, denoted by τ, is selected from $\{10, 20, \ldots, 50\}$ (hours). Moreover, the maximum variance of the workload of each task is also τ. That is, the workload of each task is randomly selected from $[0, 2\tau]$.



(a) Cambridge Haggle: Intel (b) Cambridge Haggle: Cambridge (c) Cambridge Haggle: Infocom (d) UMassDieselNet

Fig. 7. Performance comparisons on real traces: the average makespan versus the average workload of tasks (The number of tasks $m = 300$.).



(a) Cambridge Haggle: Intel (b) Cambridge Haggle: Cambridge (c) Cambridge Haggle: Infocom (d) UMassDieselNet

Fig. 8. Performance comparisons on real traces: the average makespan versus the number of tasks (The average workload of tasks $\tau = 20$.).

5.3 Synthetic Traces and Simulation Settings

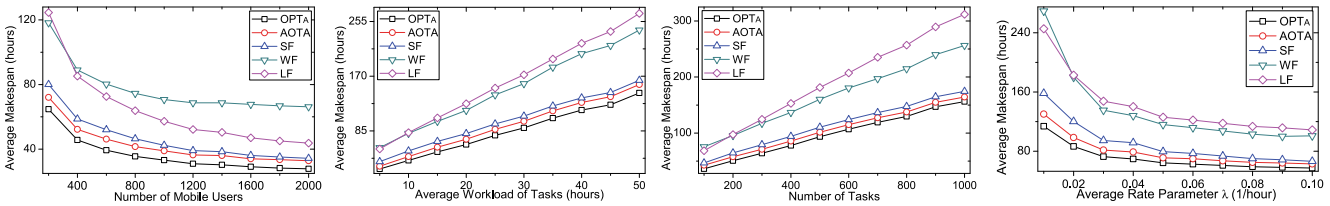
In order to evaluate the performances of our algorithms with different numbers of users and inter-meeting times, we also conduct a series of simulations on synthetic traces. First, we determine *the number of mobile users*, which is selected from $\{100, 200, \ldots, 1{,}000\}$. Then, we randomly select $5 \sim 10$ percent of these users as the requesters. Next, we determine *the average rate parameter* λ for the exponentially distributed inter-meeting time between the requesters and other users, which is selected from $\{0.01, 0.02, \ldots, 0.10\}$ (1/hour). Then, for each pair of requesters and mobile users, we randomly select a value from $[0, 2\lambda]$ as the rate parameter between them. Based on these network parameters, we construct an MSN to produce the synthetic trace. Finally, like the simulations on the real traces, we also generate the tasks for each requester, where the number of tasks is selected from $\{100, 200, \ldots, 1{,}000\}$, and the average workload is selected from $\{5, 10, \ldots, 50\}$ hours.

5.4 Evaluation Results

First, we evaluate the average makespan performances of AOTA, WF, LF, SF, and OPT_A through two groups of simulations on real traces, with different numbers of tasks and diverse average workloads. In the first group of simulations, we conduct these algorithms by changing the average workload, while keeping the number of tasks fixed. The results of the average makespans are shown in Fig. 7. In the second group of simulations, we change the number of tasks, while keeping the average workload of all tasks fixed. The results of the average makespans are shown in Fig. 8. In these simulations, the average makespan value of AOTA is about 12.2 percent larger than that of OPT_A, and the average makespan values of SF, WF, LF are about 7.86, 73.3, 86.3 percent larger than that of AOTA, respectively. That is to say, AOTA achieves the average makespan value closest to that of OPT_A, having a better performance than all other compared algorithms. This is because AOTA adopts an online task assignment strategy, which is composed of multiple rounds of virtual optimal offline task assignments. In addition, along with the increase of the average workload or the number of tasks, the absolute errors of our algorithms compared to OPT_A in both groups of simulations have little change. This is because the absolute errors of our algorithms mainly depend on the inter-meeting times between requesters and other users, and these inter-meeting times are actually derived from the real traces so that they keep unchanged in the whole simulations. Moreover, when the number of tasks and the average workload increase, the average makespans of all algorithms become larger, and the ratios of average makespans between our algorithms and OPT_A become very close to 1, since the absolute errors change little. These observations exactly validate our theoretical analysis results.

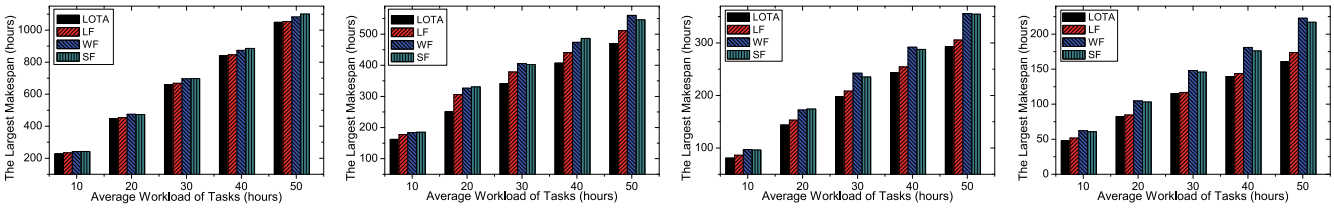
Second, we evaluate the average makespan performances of AOTA, WF, LF, SF, and OPT_A on the synthetic traces, in which we take into account the different number of users, the average workload, the number of tasks, and the average rate parameter. When we evaluate the performances for a parameter, we keep the other three parameters fixed. The results are shown in Fig. 9. The average makespan values of SF, WF, LF are about 10.2, 69.5, 70.1 percent larger than that of AOTA respectively, which also prove that AOTA has the better performance than WF, LF, and SF. As in the simulations on real traces, along with the increase of the average workload or the number of tasks, the absolute errors of our algorithm also have little change. Nevertheless, along with the increase of the average rate parameter, the absolute errors of our algorithms become smaller and smaller. This is because the average inter-meeting time decreases, which leads to a decrease in absolute errors. Additionally, when the number of mobile users increases, the average time for a requester to meet a user and the average number of tasks assigned to each user both decrease, so that the average makespans of all algorithms become smaller and smaller. Moreover, the performance gains of online task assignment decision also become smaller. As a result, the average makespan values of AOTA and SF become closer and closer, as shown in Fig. 9a.

Third, we evaluate the largest makespan performances of LOTA, LF, WF, and SF through the simulations on the real traces and the synthetic traces, respectively. For the simulations on the real traces, we change the average workload or the number of tasks, while keeping other parameters fixed.



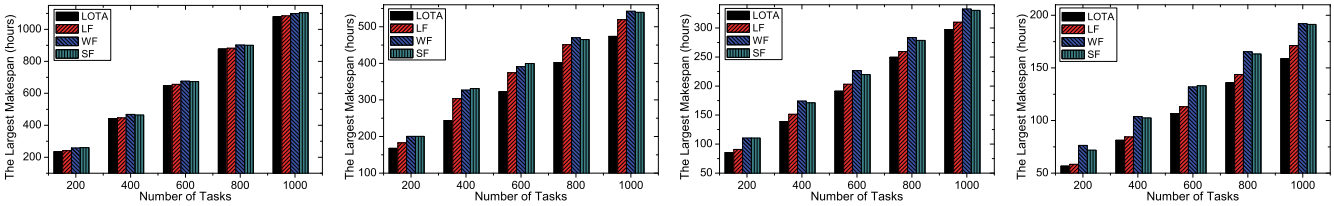
(a) Change the number of users (b) Change the average workload (c) Change the number of tasks (d) Change the average rate parameter

Fig. 9. Performance comparisons on the synthetic traces: the average makespan versus the number of users, the average workload, the number of tasks, or the average rate parameter.



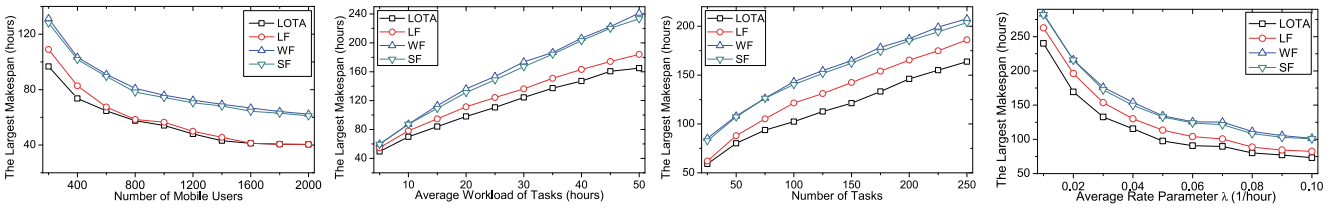
(a) Cambridge Haggle: Intel (b) Cambridge Haggle: Cambridge (c) Cambridge Haggle: Infocom (d) UMassDieselNet

Fig. 10. Performance comparisons on real traces: the largest makespan versus the average workload of tasks (The number of tasks $m = 100$.).



(a) Cambridge Haggle: Intel (b) Cambridge Haggle: Cambridge (c) Cambridge Haggle: Infocom (d) UMassDieselNet

Fig. 11. Performance comparisons on real traces: the largest makespan versus the number of tasks (The average workload of tasks $\tau = 20$.).



(a) Change the number of users (b) Change the average workload (c) Change the number of tasks (d) Change the average rate parameter

Fig. 12. Performance comparisons on the synthetic traces: the largest makespan versus the number of users, the average workload, the number of tasks, or the average rate parameter.

The results are shown in Figs. 10 and 11. For the simulations on the synthetic traces, we change the number of users, the average workload, the number of tasks, and the average rate parameter, respectively. The results are illustrated in Fig. 12. All of these results prove that LOTA has a better performance on the largest makespan than other compared algorithms, regardless if we adopt the real or synthetic traces. Here, in Fig. 12a, LOTA and LF achieve almost the same results on the largest makespans, when the number of mobile users exceeds 1,400. This is due to the reason that when the number of users is larger than the number of tasks, the number of tasks assigned to each user is very small. As a result, the online decision can only slightly improve the task assignment performance.

6 Related Work

This paper focuses on the task assignment problem in mobile crowdsensing. By far, many task allocation algorithms have been designed for mobile crowdsensing [1], [10], [11], [12], [18], [30]. For example, M. Cheung et al. in [10] formulate a movement-related task allocation problem as a task selection game, and propose a distributed algorithm for each user to select its task and determine its movement. He et al. in [12] propose a greedy approximation algorithm and a genetic algorithm for the user recruitment problem of crowdsensing in vehicular networks, where future trajectories of users are taken into account. He et al. in [18] considered the task allocation problem with the constraint of time budgets. Additionally, several incentive mechanisms are designed for crowdsensing [9], [13], [14], [15], [19]. For instance, Peng et al. in [9] propose a quality-based incentive mechanism. However, none of these existing works discussed the makespan sensitive task assignment problem in mobile crowdsensing.

On the other hand, our task assignment problem is also different from traditional parallel machine scheduling problems. In fact, there have been thousands of papers on

parallel machine scheduling problems by far. Literatures [27], [31] have made a detailed review on these works. Even in recent years, there is still much research on the complex parallel machine scheduling problems, such as [32], [33]. The most related works among the existing researches are the parallel machine scheduling algorithms that take the setup time into consideration. In these works, each task is assumed to have a different setup time, but remains identical to all machines. In contrast, the tasks being sent to mobile users in our problem follow the mobility model. More specifically, each task being sent to a mobile user is a probabilistic event. The probability distribution might be different for mobile users, but remains common for diverse tasks. Moreover, mobile users need to return the result for each task, which is also a probabilistic event. Such a unique task assignment model makes our problem different from existing parallel machine scheduling problems.

In addition, there is also much research in the MSN field, which mainly focuses on routing problems [24], [25]. None of them have studied task assignment problems, except Water Filling, which is proposed to allocate tasks among the mobile users in an MSN [26]. Although this work also takes the delivery time of the tasks and their results into consideration, it only focuses on the problem of offline task allocation for minimizing the latest makespan of all tasks, unlike ours.

7 Conclusion

In this paper, we study the MAM and MLM task assignment problems for mobile crowdsensing in MSNs, and propose two online task assignment algorithms: AOTA and LOTA. The AOTA algorithm adopts the greedy STFA and EURT strategy to assign tasks. It is applicable to the mobile crowdsensing for a group of independent sensing tasks, and can minimize the total or average makespan of all tasks. In contrast, the LOTA algorithm assigns tasks based on the greedy LTFA and EURT strategy. It is suitable to the crowdsensing for collaborative sensing tasks, and can minimize the largest makespan of all tasks. Moreover, through the theoretical analysis and a series of simulations on real traces and synthetic traces, we prove that both AOTA and LOTA can produce nearly optimal task assignment solutions.

Acknowledgments

This paper is an extended version of the conference paper [1] published in IEEE Infocom 2015. This research was supported in part by the National Natural Science Foundation of China (NSFC) (Grant No. 61572457, 61379132, 61502261, 61303206, 61572342), the Natural Science Foundation of Jiangsu Province in China (Grant No. BK20131174, BK2009150), and US National Science Foundation grants ECCS 1231461, ECCS 1128209, CNS 1138963, CNS 1065444, and CCF 1028167. Mingjun Xiao is the corresponding author.

References

[1] M. Xiao, J. Wu, L. Huang, Y. Wang, and C. Liu, "Multi-task assignment for crowdsensing in mobile social networks," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2227–2235.

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[3] V. Coric and M. Gruteser, "Crowdsensing maps of on-street parking spaces," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, 2013, pp. 115–122.

[4] A. Farshad, M. K. Marina, and F. Garcia, "Urban WiFi characterization via mobile crowdsensing," in *Proc. IEEE Netw. Operations Manage. Symp.*, 2014, pp. 1–9.

[5] Y. Zhao, et al., "CityDrive: A map-generating and speed-optimizing driving system," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1986–1994.

[6] E. Koukoumidis, L.-S. Peh, and M. Martonosi, "SignalGuru: Leveraging mobile phones for collaborative traffic signal schedule advisory," in *Proc. 9th Int. Conf. Mobile Syst. Appl. Serv.*, 2010, pp. 127–140.

[7] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "GreenGPS: A participatory sensing fuel-efficient maps application," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Serv.*, 2010, pp. 151–164.

[8] Y. Wang, W. Hu, Y. Wu, and G. Cao, "SmartPhoto: A resource-aware crowdsourcing approach for image sensing with smartphones," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2014, pp. 113–122.

[9] D. Peng, F. Wu, and G. Chen, "Pay as how well you do: A quality based incentive mechanism for crowdsensing," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2015, pp. 177–186.

[10] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed time-sensitive task selection in mobile crowdsensing," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2015, pp. 157–166.

[11] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2254–2262.

[12] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2542–2550.

[13] Q. Zhang, Y. Wen, X. Tian, X. Gan, and X. Wang, "Incentivize crowd labeling under budget constraint," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2812–2820.

[14] X. Zhang, G. Xue, R. Yu, D. Yang, and J. Tang, "Truthful incentive mechanisms for crowdsourcing," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2830–2838.

[15] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 173–184.

[16] G. Cardone, et al., "Fostering participaction in smart cities: A geo-social crowdsensing platform," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 112–119, Jun. 2013.

[17] X. Hu, T. H. S. Chu, H. C. B. Chan, and V. C. Leung, "Vita: A crowdsensing-oriented mobile cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 1, pp. 148–165, Jun. 2013.

[18] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 745–753.

[19] Y. Wei, Y. Zhu, H. Zhu, Q. Zhang, and G. Xue, "Truthful online double auctions for dynamic mobile crowdscouring," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2074–2082.

[20] Q. Zhao, Y. Zhu, H. Zhu, J. Cao, G. Xue, and B. Li, "Fair energy-efficient sensing task allocation in participatory sensing with smartphones," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1366–1374.

[21] W. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling time-variant user mobility in wireless mobile networks," in *Proc. IEEE Conf. Comput. Commun.*, 2007, pp. 758–766.

[22] L. Jeremie, F. Timur, and C. Vania, "Evaluating mobility pattern space routing for DTNs," in *Proc. IEEE Conf. Comput. Commun.*, 2006, pp. 1–10.

[23] H. Cai and D. Y. Eun, "Crossing over the bounded domain: From exponential to power-law inter-meeting time in MANET," in *Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw.*, 2007, pp. 159–170.

[24] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: A social network perspective," in *Proc. 10th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2009, pp. 299–308.

[25] J. Wu, M. Xiao, and L. Huang, "Homing spread: Community home-based multi-copy routing in mobile social networks," in *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 2319–2327.

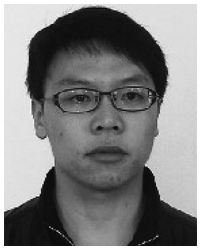
[26] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. 13th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2012, pp. 145–154.

[27] A. Allahverdi, C. Ng, T. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, 2008.
[28] J. Scott, et al., "CRAWDAD data set cambridge/ haggle (v. 2009–05-29)," May 2009. [Online]. Available: http://crawdad.cs.dartmouth.edu/cambridge/haggle
[29] J. Burgess, et al., "CRAWDAD data set umass/diesel (v. 2008–09-14)," Sep. 2008. [Online]. Available: http://crawdad.cs.dartmouth.edu/umass/diesel
[30] Q. Zhao, Y. Zhu, H. Zhu, J. Cao, G. Xue, and B. Li, "Fair energy-efficient sensing task allocation in participatory sensing with smartphones," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1366–1374.
[31] T. Cheng and C. Sin, "A state-of-the-art review of parallel-machine scheduling research," *Eur. J. Oper. Res.*, vol. 47, no. 3, pp. 271–292, 1990.
[32] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein, "Approximation techniques for average completion time scheduling," *SIAM J. Comput.*, vol. 31, no. 1, pp. 146–166, 2001.
[33] B. Alidaee and H. Li, "Parallel machine selection and job scheduling to minimize sum of machine holding cost, total machine time costs, and total tardiness costs," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 1, pp. 294–301, Jan. 2014.

Mingjun Xiao received the PhD degree from the University of Science and Technology of China (USTC), in 2004. He is an associate professor in the School of Computer Science and Technology, USTC. In 2012, he was a visiting scholar with Temple University, under the supervision of Dr. Jie Wu. He has served as a TPC member or a reviewer for many top conferences and journal papers. His main research interests include mobile computing, crowdsensing, and mobile social networks. He is a member of the IEEE.

Jie Wu is an associate vice provost of International Affairs with Temple University. He also serves as the chair and Laura H. Carnell professor in the Department of Computer and Information Sciences. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor with Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including the *IEEE Transactions on Service Computing* and the *Journal of Parallel and Distributed Computing*. He received the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award. He was a general co-chair/chair of IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as program co-chair of IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society distinguished visitor, ACM distinguished speaker, and chair of the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF distinguished speaker and a fellow of the IEEE.

Liusheng Huang received the MS degree in computer science from the University of Science and Technology of China, in 1988. He is a professor in the School of Computer Science and Technology, University of Science and Technology of China. He serves on the editorial board of many journals. He has published six books, and more than 200 papers. His main research interests include mobile computing and mobile social networks.



Ruhong Cheng received the BS degree from the School of Computer Science and Technology at the University of Science and Technology of China, in 2010. From 2013 to 2016, he was a master student of computer science at the University of Science and Technology of China, under the supervision of Dr. Mingjun Xiao. His research interests include mobile crowdsensing and mobile social networks.

Yunsheng Wang received the PhD degree in computer science from Temple University, in 2013. He is an assistant professor in the Department of Computer Science, Kettering University. His current research interests include various topics in the application and protocols of wireless networks, efficient communication in delay tolerant networks, mobile opportunistic social networks, and vehicular networks. He is the editor of the *International Journal of Ad Hoc and Ubiquitous Computing*. He is the track co-chair of ICA3PP 2016. His research is supported by the US National Science Foundation and the US Department of Justice. He is a member of the IEEE.