Crowdsourcing Task Scheduling in Mobile Social Networks

Jiahao Fan, Xinbo Zhou, Xiaofeng Gao(✉), and Guihai Chen

Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science and Engineering, Shanghai Jiao Tong University,
Shanghai 200240, China
{j.h.fan,zxb16161616}@sjtu.edu.cn, {gao-xf,gchen}@cs.sjtu.edu.cn

Abstract. With the growing popularity of mobile devices, a new paradigm called mobile crowdsourcing emerged in the recent years. Mobile users with restricted computational capability and sensing ability are now able to conduct complex tasks with the help of other users in the same mobile crowdsourcing system. In this paper, we consider the mobile crowdsourcing system model based on the spontaneously-formed mobile social networks (MSNs). We introduce two crowdsourcing task scheduling problems under this system model, with one problem aiming to minimize the operating cost of some crowdsourcing tasks and the other focusing on minimizing the overall completion time of tasks belonging to the same project. Correspondingly, under offline settings, we propose an optimal algorithm and an approximation algorithm for these two problems respectively. The optimality and the approximation ratio are analyzed accordingly. Based on these two algorithms, we further design two online algorithms to deal with the problems under online settings and their competitive ratios are computed. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive numerical experiments on synthetic datasets.

Keywords: Crowdsourcing · Task scheduling · Mobile social network

1 Introduction

With the rapid development of technology nowadays, small-sized portable mobile devices are prevailing and this enables mobile users to conduct complex tasks with the embedded powerful mobile sensors. These tasks may include air quality [4] and urban noise [11] measurement, traffic [6] and road surface [13] monitoring, etc. Since the computational capability and sensing ability of a single mobile user may not be enough to carry out a large project which consists of many relatively small and independent tasks, it is necessary to seek the help of other users and distribute these tasks among them in order to complete the whole project as efficiently as possible. This forms the idea of *mobile crowdsourcing*.

Most mobile crowdsourcing activities are carried out in a large mobile crowdsourcing system. This kind of crowdsourcing systems usually requires quite a lot

C. Pahl et al. (Eds.): ICSOC 2018, LNCS 11236, pp. 317–331, 2018.
https://doi.org/10.1007/978-3-030-03596-9_22

of human and material resources for management and maintenance. If a requester just wants to distribute the crowdsourcing tasks through his social relationships (including families, friends, colleagues, acquaintances, etc.), the requester may conduct his own crowdsourcing activities by taking the advantage of the existing *mobile social networks* (MSNs) to avoid unnecessary overheads and thus save the budget. This idea seems more appealing than the traditional crowdsourcing paradigm to those requesters with limited budgets.

In this paper, we mainly focus on the task scheduling problems for MSN-based mobile crowdsourcing systems. Unlike traditional task scheduling problems in mobile crowdsourcing, we require the requester and the crowd workers to actually make person-to-person contacts, which are some probabilistic events in the mobile social networks, so that crowdsourcing tasks can be sent by the requester and their feedbacks can be delivered by the crowd workers. Each crowdsourcing task requires one contact between the requester and the crowd worker when it is being distributed and another when its feedback is being delivered. Our objectives are strongly related to the completion time of each task, which consists of the time for its distribution, processing, and feedback delivery.

Suppose there is an operating cost related to task s_i when it is being scheduled and processed in the mobile crowdsourcing system, and the cost is proportional to the completion time of s_i. Then, the cost of s_i can be denoted as $w_i C_i$, where w_i is the proportion of the cost to the task completion time, and C_i is the actual completion time of s_i. Our min-WCT problem aims to minimize the total weighted completion time of all the crowdsourcing tasks, so that the total cost of conducting the crowdsourcing campaign can be minimized. Meanwhile, in some scenarios, the crowdsourcing tasks of a requester may be part of a larger project, and all of them should be completed as soon as possible. Hence, we introduce the min-MCT problem to minimize the maximum completion time of all the crowdsourcing tasks (i.e., $\max C_i$) in order to accelerate the progress of the following stages in the project.

More specifically, the main contributions of this paper include:

1. We introduce two task scheduling problems for crowdsourcing in mobile social networks and formulate them as the min-WCT problem and the min-MCT problem respectively.
2. For the min-WCT problem, we propose an optimal offline algorithm named LWF and give the proof of its optimality. Based on LWF, we design an online algorithm named CosMOS to deal with min-WCT under online settings and give the analysis of its competitive ratio.
3. For the min-MCT problem, we propose an offline approximation name LRSTF and give the analysis of its approximation ratio. Based on LRSTF, we also design another online algorithm named TiMOS to deal with min-MCT under online settings and give the analysis of its competitive ratio.
4. We conduct extensive numerical experiments on synthetic datasets to compare our algorithms with some traditional scheduling algorithms. Both the theoretical analysis and the experimental results validate the effectiveness and efficiency of our designs.

The rest of this paper is organized as follows. Section 2 discusses some related works. Section 3 describes the MSN-based mobile crowdsourcing system model. In Sect. 4, we formulate the min-WCT problem, and propose an optimal offline solution named LWF and an online solution named CosMOS. In Sect. 5, we formulate the min-MCT problem, and introduce an offline approximation named LRSTF and an online solution named TiMOS. In Sect. 6, we conduct numerical experiments to evaluate the performance of our designs. Section 7 is the final conclusion.

2 Related Work

Mobile crowdsourcing is getting more and more research interests in the recent years. Some of the literatures focus on the framework and application design [9,18], while others pay more attention to the specific stages in the process of mobile crowdsourcing, such as task scheduling [16], incentive mechanisms [5], quality control [17], and security and privacy issues [15]. Generally, there are three components or participants in a typical mobile crowdsourcing system (MCS): requesters or crowdsourcers, crowd workers, and crowdsourcing platform [14]. Specifically, to crowdsource a task, a requester submits the task to a crowdsourcing platform, and optionally, after receiving the solutions provided by crowd workers, rates their qualities. Crowd workers choose to work on those tasks and attempt to submit their solutions as feedback. An intermediation platform (i.e., crowdsourcing platform) builds a link between the requesters and workers, which serves as a crowdsourcing enabler and has some rules for the whole lifecycle of crowdsourcing, such as the skill-set, certification level, due date, expected outcomes, and payments for the crowd workers.

With the growing popularity of mobile smart devices, the concept of the mobile social networks (MSNs) gradually comes into our view. As an application scenario, MSN is widely adopted in all kinds of problems including communication [19], social community detection [10], etc. Besides, researchers are also interested in exploring inner features and characteristics regarding MSN itself. In this paper, we focus on the task scheduling problems in MSN-based mobile crowdsourcing systems. There have been plenty of previous literatures concerning the topic of task scheduling problems [1–3]. Our problems may be similar to the traditional scheduling problems on parallel machines, but our system model includes a probabilistic event which is the person-to-person contact between the requester and the crowd workers. In order to solve the problems, we have to make online task scheduling decisions, and this makes our designs more realistic and applicable under the scenario of crowdsourcing in mobile social networks.

3 System Model

In this section, we introduce the participants in the mobile social network (MSN) and how they interact with each other under the paradigm of crowdsourcing.

Here, the mobile social network model in [16] is adopted. Consider a crowd of mobile users in the network, denoted as $U = \{u_0, u_1, \ldots, u_m\}$. Each of these mobile users carries a mobile device which supports wireless communication and possesses certain processing capability to carry out some crowdsourcing tasks. Two mobile users can communicate with each other only if they get close enough for Bluetooth to work, or enter the communication range of some access points respectively to interact indirectly via WiFi. Assume that the connection duration and bandwidth are enough to satisfy our needs to communicate for the purpose of crowdsourcing. We further adopt the mobility model in [7], where the pairwise inter-contact time is exponentially distributed. This means that the inter-meeting time between any pair of mobile users u_i and u_j $(i \neq j)$ follows the exponential distribution with parameter λ_{ij}. This parameter can be estimated from historical communication records between u_i and u_j.

Under the paradigm of crowdsourcing, there is a user in the MSN that hopes to recruit other mobile users to help complete some crowdsourcing tasks. The user with this kind of need is called the *requester*, and those who have the potential to carry out these tasks are called *crowd workers*. Without loss of generality, we assume u_0 is a requester and the other m users $\{u_1, u_2, \ldots, u_m\}$ in the MSN are crowd workers that are encouraged to participate in the crowdsourcing campaign by some incentive mechanisms. Since crowd workers are not supposed to communicate with each other, we denote λ_{0j} as λ_j for simplicity.

Suppose that the requester u_0 has n indivisible crowdsourcing tasks, denoted as $S = \{s_1, s_2, \cdots, s_n\}$. The workload of task $s_i \in S$ is represented by its *Required Service Time* (RST), denoted as τ_i. Each crowdsourcing task should be assigned to only one crowd worker by the requester. The distribution, as well as the feedback, of each task requires one contact between the designated crowd worker and the requester. In practice, it is more efficient for the requester to send a batch of crowdsourcing tasks to a crowd worker at their first contact and receive the feedbacks whenever they are ready at the time of their following contacts. We use $\Lambda = \{S_1, S_2, \ldots, S_m\}$ to denote an scheduling decision of the n crowdsourcing tasks to the m crowd workers, where $\bigcup_{S_j \in \Lambda} S_j = S$ and $S_{j_1} \cap S_{j_2} = \phi (j_1 \neq j_2)$.

A crowd worker will process the tasks assigned to him one by one until there is none left. Thus, the *Completion Time* (CT) of task s_i includes three parts: (a) the time for the requester u_0 and the crowd worker u_j, to whom we assign the task s_i (i.e., $s_i \in S_j$), to meet for the first time and complete the task distribution process (this includes the distribution of all the tasks in S_j), (b) the time for u_j to process all the tasks prior to s_i in S_j and task s_i itself, and (c) the time for u_0 and u_j to meet again after the feedback of s_i becomes available.

Since we cannot foresee the exact time when the requester and the crowd workers meet, we define the *Expected Meeting Time* (EMT), which is $\frac{1}{\lambda_j}$ under the assumption of the exponentially distributed inter-meeting time between u_0 and u_j, to represent the time in part (a) and part (c). Therefore, the completion time of task s_i, which is denoted as C_i, consists of two expected meeting time intervals and the time for the crowd worker u_j to process all the tasks prior to s_i in S_j and task s_i itself.

4 Cost Minimized Scheduling

In this section, we first formulate the cost-related min-WCT problem for crowdsourcing task scheduling. Then, we propose an optimal algorithm LWF to solve the problem under offline settings, followed by the proof of its optimality. Finally, we give an online algorithm CosMOS based on LWF to solve the problem under online settings together with the analysis of its competitive ratio.

4.1 Problem Formulation

Consider the system model described in the previous section (Sect. 3). Assume that all the tasks have the same required service time, i.e., $\tau_1 = \tau_2 = \cdots = \tau_n$. Suppose there is an operating cost related to task s_i when it is being scheduled and processed in the crowdsourcing system, and the cost is proportional to the completion time of task s_i. Our goal is to minimize the total operating cost of carrying out all the tasks in S. This is equivalent to the following definition of the min-WCT problem.

Definition 1 (Minimizing the Weighted Completion Time (min-WCT)). *Given tasks $S = \{s_1, s_2, \ldots, s_n\}$ with the same required service time (i.e., $\tau_1 = \tau_2 = \cdots = \tau_n$) and their corresponding weights $W = \{w_1, w_2, \ldots, w_n\}$, min-WCT aims to find an scheduling decision $\Lambda = \{S_1, S_2, \ldots, S_m\}$ among crowd workers $U = \{u_1, u_2, \ldots, u_m\}$ such that the total weighted completion time of all the tasks $\sum_{i=1}^{n} w_i C_i$ is minimized.*

Denote $\sum_{i=1}^{n} w_i C_i$ as WCT in this section for simplicity.

4.2 Offline Task Scheduling

First, we would like to consider the min-WCT problem under offline settings. In this scenario, the requester is supposed to make the scheduling decision before any contacts with the potential crowd workers and stick to this decision throughout the entire crowdsourcing campaign.

Intuitively, we wish to reduce the completion time for tasks with large weights so that we can reduce the total weighted completion time significantly. Following this idea, we design the *Largest Weight First* (LWF) algorithm (Algorithm 1) to solve the min-WCT problem under offline settings. The concept of a crowd worker's *Expected Workload* (EW) is defined as follows.

Definition 2 (Expected Workload (EW)). *The expected workload EW_j of a crowd worker u_j consists of three parts: (a) the expected meeting time for u_j and the requester u_0 to meet for the first time and complete the task distribution process, (b) the total required service time of all the tasks assigned to u_j, i.e., the tasks in S_j, and (c) the expected meeting time for u_j and u_0 to meet again and deliver the feedback of the tasks. Furthermore, we define $EW_j = \frac{2}{\lambda_j}$ if no task is assigned to u_j and S_j is empty.*

Algorithm 1: The LWF Algorithm

Input: $S = \{s_1, s_2, \ldots, s_n : \tau_1 = \tau_2 = \cdots = \tau_n\}$,
$W = \{w_1, w_2, \ldots, w_n : w_1 \geq w_2 \geq \cdots \geq w_n\}$,
$U = \{u_1, u_2, \ldots, u_m : \lambda_1, \lambda_2, \ldots, \lambda_m\}$
Output: $\Lambda_{\text{LWF}} = \{S_1, S_2, \ldots, S_m\}$

```
for j ← 1 to m do
    S_j ← φ;
    EW_j ← 2/λ_j;
for i ← 1 to n do
    j_min ← arg min{EW_k | u_k ∈ U};
    S_{j_min} ← S_{j_min} ∪ {s_i};
    EW_{j_min} ← EW_{j_min} + τ_i;
return Λ_LWF = {S_1, S_2, ..., S_m};
```

The Optimality. We briefly describe the proof of its optimality as follows. Assume $\Lambda_{\text{opt}} = \{S_1^*, S_2^*, \ldots, S_m^*\}$ is the optimal scheduling decision for the min-WCT problem under offline settings.

Lemma 1. *For any $S_j^* \in \Lambda_{opt}$, a task $s_{j_p} \in S_j^*$ with a larger weight w_{j_p} is processed prior to the processing of another task $s_{j_q} \in S_j^* \backslash \{s_{j_p}\}$ with a smaller weight w_{j_q} (ties may be broken arbitrarily).*

According to Lemma 1, we can conclude that in order to achieve the optimal solution, tasks assigned to the same crowd worker must be processed in a non-increasing order according to their weights by this worker.

Lemma 2. *Suppose that we have somehow determined an scheduling decision for $\{s_1, s_2, \ldots, s_{i-1}\}$ and this partial scheduling shall not be further changed, which means all the other tasks must be scheduled behind them, the optimal solution to the min-WCT problem under this condition must assign task s_i to the crowd worker with the currently smallest expected workload, denoted as u_j.*

Then, we are ready to prove the optimality of the LWF algorithm as follows.

Theorem 1. *As for the min-WCT problem, the LWF algorithm achieves the optimal solution by assigning the task with the largest weight to the crowd worker with the currently smallest expected workload in each round until all tasks in S have been assigned.*

Proof. We prove this theorem by induction. The base case where no task has been assigned is trivial. For the inductive step, assume we can obtain the optimal scheduling decision for tasks in $\{s_1, s_2, \ldots, s_{k-1}\}(1 \leq k \leq n)$ by the LWF algorithm. Then, according to Lemmas 1 and 2, we can also obtain the optimal scheduling decision for tasks in $\{s_1, s_2, \ldots, s_k\}$ through another round of scheduling by the LWF algorithm. Finally, we can obtain the optimal scheduling decision for all the tasks in $S = \{s_1, s_2, \ldots, s_n\}$ from the output of the LWF algorithm. This finishes the proof of Theorem 1.

4.3 Online Task Scheduling

Next, we would like to consider the min-WCT problem under online settings. In this scenario, the requester does not need to make the final scheduling decision at first. Instead, he can adjust his decision with the progress of the crowdsourcing campaign. Since the requester gets a better knowledge of the crowd whenever he meets a crowd worker, he can make the scheduling decision regarding this crowd worker at the time of their first contact to improve the final result.

We build the *Cost Minimized Online Scheduing* (CosMOS) algorithm (Algorithm 2) based on our previous design. In the CosMOS algorithm, we make the scheduling decision regarding each crowd worker at the time of his first contact with the requester. When u_0 meets u_j, we determine a partial scheduling decision of $S_j = \{s_{j_1}, s_{j_2}, \ldots, s_{j_k}\}$ and make it final. We denote the process of determining S_j as *decision step* DS_j in the CosMOS algorithm.

Algorithm 2: The CosMOS Algorithm

Input: $S = \{s_1, s_2, \ldots, s_n : \tau_1 = \tau_2 = \cdots = \tau_n\}$,
$W = \{w_1, w_2, \ldots, w_n : w_1 \geq w_2 \geq \cdots \geq w_n\}$,
$U = \{u_1, u_2, \ldots, u_m : \lambda_1, \lambda_2, \ldots, \lambda_m\}$
Output: $\Lambda_{\text{CosMOS}} = \{S_1, S_2, \ldots, S_m\}$

when u_0 *meets* u_j **do**
  $S_j \leftarrow \phi$;
  $EW_j \leftarrow \frac{1}{\lambda_j}$;
  foreach k *such that* $u_k \in U \backslash \{u_j\}$ **do**
    $S_k \leftarrow \phi$;
    $EW_k \leftarrow \frac{2}{\lambda_k}$;
  foreach i *such that* $s_i \in S$ *(in an increasing order of* i*)* **do**
    $k_{min} \leftarrow \arg\min\{EW_k \mid u_k \in U\}$;
    $S_{k_{min}} \leftarrow S_{k_{min}} \cup \{s_i\}$;
    $EW_{k_{min}} \leftarrow EW_{k_{min}} + \tau_i$;
  $S \leftarrow S \backslash S_j$;
  $U \leftarrow U \backslash \{u_j\}$;
return $\Lambda_{CosMOS} = \{S_1, S_2, \ldots, S_m\}$;

Performance Analysis. Without loss of generality, we assume that the requester u_0 meets the crowd workers in the order of $u_1, u_2, \ldots, u_m$. Assume that after decision step DS_j, the total weighted completion time is WCT_j. Obviously, $WCT_{\text{CosMOS}} = WCT_m$. Furthermore, let $WCT_0 = WCT_{\text{LWF}}$.

Theorem 2. $WCT_{LWF} = WCT_0 \geq WCT_1 \geq \cdots \geq WCT_m = WCT_{CosMOS}$.

Proof. In any decision step $DS_j (1 \leq j \leq m)$, we will not change the scheduling of any of the tasks in $\bigcup_{1 \leq i \leq j-1} S_i$, and the total weighted completion time of

these tasks remains unchanged before and after the decision step. As for the remaining tasks, we can prove that the modified LWF algorithm achieves the optimal solution under the condition that u_0 has met u_j by duplicating the proof of the optimality of our LWF algorithm. Therefore, we have $WCT_{j-1} \geq WCT_j$, and this finishes the proof of Theorem 2.

Now, we give the competitive ratio of the CosMOS algorithm as follows.

Theorem 3. *Assume someone can foresee the mobilities of all the crowd workers, so that he knows exactly at what time each meeting between the requester and the crowd workers will happen. Based on this knowledge, he can give an optimal online task scheduling decision, denoted as* $\Lambda_{OPT} = \{S_1^*, S_2^*, \ldots, S_m^*\}$. *Then, we have*

$$\frac{WCT_{CosMOS}}{WCT_{OPT}} \leq 1 + \frac{w_{max} \sum_{j=1}^{m} \frac{2}{\lambda_j}}{w_{min} \tau_{min}}.$$

Proof. First, we can give WCT_{OPT} as follows.

$$WCT_{\mathrm{OPT}} = \sum_{j=1}^{m} \sum_{s_{j_k} \in S_j^*} w_{j_k}(t_j + t'_j + \mathcal{T}_{j_k} + \tau_{j_k}).$$

If we adopt the scheduling decision of Λ_{OPT} in the offline version of min-WCT and denote it as Λ', then we have

$$\begin{aligned} WCT' &= \sum_{j=1}^{m} \sum_{s_{j_k} \in S_j^*} w_{j_k}(\frac{2}{\lambda_j} + \mathcal{T}_{j_k} + \tau_{j_k}) \\ &= WCT_{\mathrm{OPT}} + \sum_{j=1}^{m} \sum_{s_{j_k} \in S_j^*} w_{j_k}(\frac{2}{\lambda_j} - t_j - t'_j) \\ &\leq WCT_{\mathrm{OPT}} + \sum_{j=1}^{m} \sum_{s_{j_k} \in S_j^*} w_{j_k} \cdot \frac{2}{\lambda_j}. \end{aligned}$$

Since Λ_{LWF} is the optimal scheduling for the offline version of min-WCT, then we have $WCT_{\mathrm{LWF}} \leq WCT'$. Combined with Theorem 2, we can get

$$WCT_{\mathrm{CosMOS}} \leq WCT_{\mathrm{LWF}} \leq WCT' \leq WCT_{\mathrm{OPT}} + \sum_{j=1}^{m} \sum_{s_{j_k} \in S_j^*} w_{j_k} \cdot \frac{2}{\lambda_j}.$$

The lower bound of WCT_{OPT} should be $n \cdot w_{min} \cdot \tau_{min}$, and this gives us

$$\begin{aligned} \frac{WCT_{\mathrm{CosMOS}}}{WCT_{\mathrm{OPT}}} &\leq 1 + \frac{\sum_{j=1}^{m} \sum_{s_{j_k} \in S_j^*} w_{j_k} \cdot \frac{2}{\lambda_j}}{WCT_{\mathrm{OPT}}} \\ &\leq 1 + \frac{w_{max} \sum_{j=1}^{m} \frac{2}{\lambda_j}}{w_{min} \tau_{min}}. \end{aligned}$$

Thus, Theorem 3 holds.

5 Time Minimized Scheduing

In this section, we first formulate the time-related min-MCT problem for task scheduling in mobile crowdsourcing systems. Then, we propose an approximation algorithm LRSTF to solve the problem under offline settings, followed by the performance analysis regarding its approximation ratio. Finally, we give an online algorithm TiMOS based on LRSTF to solve the problem under online settings together with the analysis of its competitive ratio.

5.1 Problem Formulation

In some cases, the crowdsourcing tasks of a requester may be part of a larger project, and all of them should be completed as soon as possible. Hence, we introduce the min-MCT problem to minimize the maximum completion time of all these tasks in order to accelerate the progress of the following stages in the project. In this problem, crowdsourcing tasks now have different values of the required service time and there is no weight related to any of these tasks. Our goal is equivalent to minimizing the completion time of the latest completed task. The problem is formally defined as follows.

Definition 3 (Minimizing the Maximum Completion Time (min-MCT)). *Given tasks* $S = \{s_1, s_2, \ldots, s_n\}$ *with required service time* τ_i *related to task* s_i*, min-MCT aims to find an scheduling decision* $\Lambda = \{S_1, S_2, \ldots, S_m\}$ *among crowd workers* $U = \{u_1, u_2, \ldots, u_m\}$ *such that the maximum completion time of all the tasks* $\max\limits_{1 \le i \le n} C_i$ *is minimized.*

Denote $\max\limits_{1 \le i \le n} C_i$ as MCT in this section for simplicity.

The NP-Hardness. Consider a special case of the min-MCT problem, in which there may exist one requester and two crowd workers in the mobile crowdsourcing system. The expected inter-meeting time between the requester and any one of the crowd workers is assumed to be zero. In this case, the min-MCT problem is equivalent to the *PARTITION* problem, which is traditionally considered to be NP-hard according to [8]. Therefore, the general case of the min-MCT problem is also NP-hard.

5.2 Offline Task Scheduling

The observation we have on the min-MCT problem is that we should leave the tasks with relatively short required service time to the end of our scheduling process so that the expected workloads of all the workers can be balanced and the maximum completion time of all the tasks can be reduced. Unfortunately, this idea only leads to an approximation rather than an optimal solution, which we denote as the *Longest Required Service Time First* (LRSTF) algorithm (Algorithm 3). We claim that its approximation ratio is $(\frac{3}{2} - \frac{1}{2m})$ and omit its detailed proof due to limited space.

Algorithm 3: The LRSTF Algorithm

Input: $S = \{s_1, s_2, \ldots, s_n : \tau_1 \geq \tau_2 \geq \cdots \geq \tau_n\}$,
$U = \{u_1, u_2, \ldots, u_m : \lambda_1, \lambda_2, \ldots, \lambda_m\}$
Output: $\Lambda_{\text{LRSTF}} = \{S_1, S_2, \ldots, S_m\}$

for $j \leftarrow 1$ **to** m **do**
  $S_j \leftarrow \phi$;
  $EW_j \leftarrow \frac{2}{\lambda_j}$;
for $i \leftarrow 1$ **to** n **do**
  $j_{min} \leftarrow \arg\min\{EW_k \mid u_k \in U\}$;
  $S_{j_{min}} \leftarrow S_{j_{min}} \cup \{s_i\}$;
  $EW_{j_{min}} \leftarrow EW_{j_{min}} + \tau_i$;
return $\Lambda_{LRSTF} = \{S_1, S_2, \ldots, S_m\}$;

5.3 Online Task Scheduling

Similar to our design of the CosMOS algorithm, we build the *Time Minimized Online Scheduing* (TiMOS) algorithm (Algorithm 4) based on the LRSTF algorithm. In the TiMOS algorithm, we make the scheduling decision regarding each crowd worker at the time of his first contact with the requester. When u_0 meets u_j, we determine a partial scheduling decision of $S_j = \{s_{j_1}, s_{j_2}, \ldots, s_{j_k}\}$ and make it final. We also denote the process of determining S_j as *decision step* DS_j in the TiMOS algorithm.

Algorithm 4: The TiMOS Algorithm

Input: $S = \{s_1, s_2, \ldots, s_n : \tau_1 \geq \tau_2 \geq \cdots \geq \tau_n\}$,
$U = \{u_1, u_2, \ldots, u_m : \lambda_1, \lambda_2, \ldots, \lambda_m\}$
Output: $\Lambda_{\text{TiMOS}} = \{S_1, S_2, \ldots, S_m\}$

when u_0 *meets* u_j **do**
  $S_j \leftarrow \phi$;
  $EW_j \leftarrow \frac{1}{\lambda_j}$;
  foreach k *such that* $u_k \in U \backslash \{u_j\}$ **do**
    $S_k \leftarrow \phi$;
    $EW_k \leftarrow \frac{2}{\lambda_k}$;
  foreach i *such that* $s_i \in S$ *(in an increasing order of* i*)* **do**
    $k_{min} \leftarrow \arg\min\{EW_k \mid u_k \in U\}$;
    $S_{k_{min}} \leftarrow S_{k_{min}} \cup \{s_i\}$;
    $EW_{k_{min}} \leftarrow EW_{k_{min}} + \tau_i$;
  $S \leftarrow S \backslash S_j$;
  $U \leftarrow U \backslash \{u_j\}$;
return $\Lambda_{TiMOS} = \{S_1, S_2, \ldots, S_m\}$;

Performance Analysis. Without loss of generality, we assume that the requester u_0 meets the crowd workers in the order of $u_1, u_2, \ldots, u_m$. First, we need to calculate the difference between the solution given by LRSTF and the optimal offline solution in each decision step. Assume that $\Lambda_{\text{opt}}^{(j)}$ is the optimal offline solution in decision step DS_j, and $\Lambda_{\text{LRSTF}}^{(j)}$ is the solution given by the LRSTF algorithm. Obviously, Λ_{TiMOS} is $\Lambda_{\text{LRSTF}}^{(m)}$, and we define $\Lambda_{\text{LRSTF}}^{(0)}$ to be Λ_{LRSTF}. Then we have the following theorem.

Theorem 4. $MCT_{opt}^{(j)} \leq MCT_{LRSTF}^{(j)} \leq MCT_{opt}^{(j)} + \tau_{max}$.

Proof. First, since $\Lambda_{\text{opt}}^{(j)}$ is the optimal offline solution in decision step DS_j, we have $MCT_{\text{opt}}^{(j)} \leq MCT_{\text{LRSTF}}^{(j)}$. Combining the two inequalities below,

$$MCT_{\text{LRSTF}}^{(j)} \leq \tau_n(1 - \frac{1}{m}) + \frac{\sum_{i=1}^{n} \tau_i + \sum_{j=1}^{m} \frac{2}{\lambda_j}}{m}$$

$$MCT_{\text{opt}}^{(j)} \geq \frac{\sum_{i=1}^{n} \tau_i + \sum_{j=1}^{m} \frac{2}{\lambda_j}}{m}$$

we can get that

$$MCT_{\text{LRSTF}}^{(j)} \leq MCT_{\text{opt}}^{(j)} + \tau_n(1 - \frac{1}{m}) \leq MCT_{\text{opt}}^{(j)} + \tau_{max}.$$

This finishes the proof of this theorem.

Similar to Theorem 2, we directly give the following theorem regarding $\Lambda_{\text{opt}}^{(j)}$.

Theorem 5. $MCT_{opt}^{(0)} \geq MCT_{opt}^{(1)} \geq \cdots \geq MCT_{opt}^{(m)}$.

Now, we can compute the competitive ratio of the TiMOS algorithm.

Theorem 6. *Assume someone can foresee the mobilities of all the crowd workers, so that he knows exactly at what time each meeting between the requester and the crowd workers will happen. Based on this knowledge, he can give an optimal online task scheduling decision, denoted as* $\Lambda_{OPT} = \{S_1^*, S_2^*, \ldots, S_m^*\}$. *Then, we have*

$$\frac{MCT_{TiMOS}}{MCT_{OPT}} \leq 2 + \frac{2}{\lambda_{min}\tau_{max}}.$$

Proof. First, we can give MCT_{OPT} as follows.

$$MCT_{\text{OPT}} = \max_{S_j^* \in \Lambda_{\text{OPT}}} \left\{ t_j + t_j' + \sum_{s_{j_k} \in S_j^*} \tau_{j_k} \right\}.$$

If we adopt the scheduling decision of Λ_{OPT} in the offline version of min-MCT and denote it as Λ', then we have

$$\begin{aligned} MCT' &= \max_{S_j^* \in \Lambda'} \left\{ \frac{2}{\lambda_j} + \sum_{s_{j_k} \in S_j^*} \tau_{j_k} \right\} \\ &\leq MCT_{\mathrm{OPT}} + \max \left\{ \frac{2}{\lambda_j} - t_j - t'_j \right\} \\ &\leq MCT_{\mathrm{OPT}} + \frac{2}{\lambda_{min}}. \end{aligned}$$

Since $\Lambda_{\mathrm{OPT}}^{(0)}$ is the optimal scheduling for the offline version of min-MCT, we have $MCT_{\mathrm{OPT}}^{(0)} \leq MCT'$. Combined with Theorems 4 and 5, we can get

$$MCT_{\mathrm{TiMOS}} = MCT_{\mathrm{LRSTF}}^{(m)} \leq MCT_{\mathrm{opt}}^{(m)} + \tau_{max} \leq MCT_{\mathrm{opt}}^{(0)} + \tau_{max} \leq MCT' + \tau_{max}.$$

Therefore, we have

$$MCT_{\mathrm{TiMOS}} \leq MCT_{\mathrm{OPT}} + \frac{2}{\lambda_{min}} + \tau_{max},$$

and further considering the fact that $MCT_{\mathrm{OPT}} \geq \tau_{max}$, we have

$$\frac{MCT_{\mathrm{TiMOS}}}{MCT_{\mathrm{OPT}}} \leq 1 + \frac{\frac{2}{\lambda_{min}} + \tau_{max}}{\tau_{max}} = 2 + \frac{2}{\lambda_{min}\tau_{max}}.$$

Thus, Theorem 6 holds.

6 Evaluation

In this section, we carry out extensive numerical experiments to evaluate the performances of the proposed algorithms.

6.1 Algorithms in Comparison

The first scheduling algorithm we would like to compare with is the Water Filling (WF) algorithm described in [12] in the context of mobile computing. More specifically, the WF algorithm assigns tasks to the earliest idle worker in their initial order, which is also called the natural order or the list order.

For the min-WCT problem, we also implement the *Smallest Weight First* (SWF) algorithm to assign tasks in a reverse order of the LWF algorithm. For the min-MCT problem, we compare our design with another algorithm called the *Shortest Required Service Time First* (SRSTF) algorithm to assign tasks in a reverse order of the LRSTF algorithm. The SRSTF algorithm is the optimal algorithm when we consider the problem of reducing the average completion time of all the tasks, and its optimality is proved in [16].

6.2 Simulation Results

Figure 1 shows the simulation results when we change the number of crowd workers, the average inter-meeting time between the requester and crowd workers, the number of tasks and the average workload of tasks respectively. Given the same conditions, LWF can always deliver a better result than WF or SWF. Meanwhile, SWF performs even worse than WF.

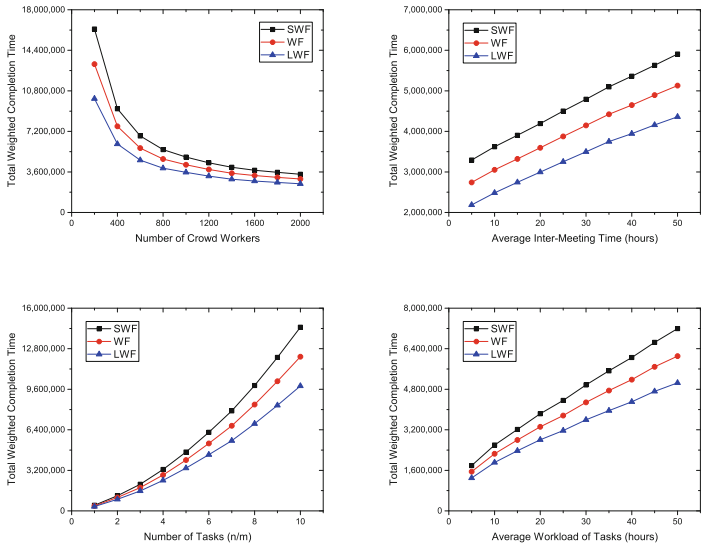

Fig. 1. Performance comparisons for min-WCT on synthetic datasets

Figure 2 shows the simulation result when we change the number of crowd workers, the average inter-meeting time between the requester and crowd workers, the number of tasks and the average workload of tasks respectively. Given the same conditions, LRSTF can always deliver a better result than WF or SRSTF. Meanwhile, SRSTF performs almost as badly as WF.

7 Conclusion

In this paper, we discuss two problems regarding task scheduling for crowdsourcing in mobile social networks. The first one is the min-WCT problem, which aims to minimize the total weighted completion time of some crowdsourcing tasks. We propose an optimal algorithm named LWF to solve this problem under offline settings and give the proof of its optimality. Based on this optimal offline algorithm, we further design an online algorithm named CosMOS to deal with the min-WCT problem under online settings. By computing the competitive ratio,

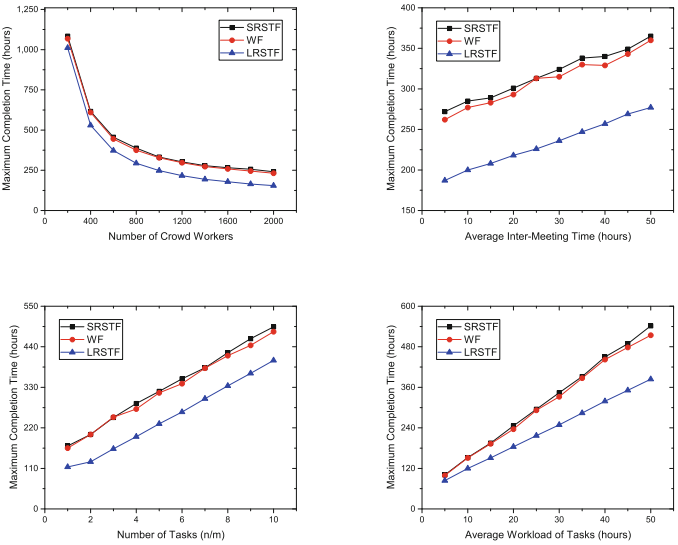

Fig. 2. Performance comparisons for min-MCT on synthetic datasets

we find that CosMOS can achieve near-optimal performance under certain circumstances. The other problem in discussion is the min-MCT problem, which focuses on minimizing the maximum completion time among the tasks belonging to the same project. Since the offline version of this problem is proved to be NP-hard, we propose an approximation algorithm named LRSTF to solve it and the approximation ratio is analyzed. Similarly, we design another online algorithm named TiMOS to deal with the online version of the min-MCT problem and its difference with the optimal solution is bounded by a fixed value. At last, we conduct extensive simulations on synthetic datasets to demonstrate the performance of our algorithms. Both the theoretical analysis and the simulation results validate the effectiveness and efficiency of our designs.

Acknowledgements. This work is supported by the National Key R&D Program of China (2018YFB1004703), the National Natural Science Foundation of China (61872238, 61672353), the Shanghai Science and Technology Fund (17510740200), the CCF-Tencent Open Research Fund (RAGR20170114), and Huawei Innovation Research Program (HO2018085286).

References

1. Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. Eur. J. Oper. Res. **187**(3), 985–1032 (2008)
2. Bridi, T., Bartolini, A., Lombardi, M., Milano, M., Benini, L.: A constraint programming scheduler for heterogeneous high-performance computing machines. IEEE Trans. Parallel Distrib. Syst. (TPDS) **27**(10), 2781–2794 (2016)

3. Ding, J., Song, S., Zhang, R., Chiong, R., Wu, C.: Parallel machine scheduling under time-of-use electricity prices: new models and optimization approaches. IEEE Trans. Autom. Sci. Eng. **13**(2), 1138–1154 (2016)
4. Dutta, P., et al.: Common sense: participatory urban sensing using a network of handheld air quality monitors. In: ACM International Conference on Embedded Networked Sensor Systems (SenSys), pp. 349–350 (2009)
5. Fan, Y., Sun, H., Liu, X.: Poster: TRIM: a truthful incentive mechanism for dynamic and heterogeneous tasks in mobile crowdsensing. In: ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), pp. 272–274 (2015)
6. Farkas, K., Nagy, A.Z., Tomas, T., Szabó, R.: Participatory sensing based real-time public transport information service. In: IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom), pp. 141–144 (2014)
7. Gao, W., Li, Q., Zhao, B., Cao, G.: Multicasting in delay tolerant networks: a social network perspective. In: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 299–308 (2009)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
9. Guo, B., Liu, Y., Wu, W., Yu, Z., Han, Q.: Activecrowd: a framework for optimized multitask allocation in mobile crowdsensing systems. IEEE Trans. Hum.-Mach. Syst. **47**(3), 392–403 (2017)
10. Kim, P., Kim, S.: A detection of overlapping community in mobile social network. In: ACM Symposium on Applied Computing (SAC), pp. 175–179 (2014)
11. Rana, R.K., Chou, C.T., Kanhere, S.S., Bulusu, N., Hu, W.: Ear-phone: an end-to-end participatory urban noise mapping system. In: International Conference on Information Processing in Sensor Networks (IPSN), pp. 105–116 (2010)
12. Shi, C., Lakafosis, V., Ammar, M.H., Zegura, E.W.: Serendipity: enabling remote computing among intermittently connected mobile devices. In: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 145–154 (2012)
13. Singh, G., Bansal, D., Sofat, S., Aggarwal, N.: Smart patrolling: an efficient road surface monitoring using smartphone sensors and crowdsourcing. Pervasive Mob. Comput. **40**, 71–88 (2017)
14. Wang, Y., Jia, X., Jin, Q., Ma, J.: Mobile crowdsourcing: framework, challenges, and solutions. Concurr. Comput.: Pract. Exp. **29**(3), 1–17 (2017)
15. Wu, S., Wang, X., Wang, S., Zhang, Z., Tung, A.K.H.: K-anonymity for crowdsourcing database. IEEE Trans. Knowl. Data Eng. (TKDE) **26**(9), 2207–2221 (2014)
16. Xiao, M., Wu, J., Huang, L., Wang, Y., Liu, C.: Multi-task assignment for crowdsensing in mobile social networks. In: IEEE Conference on Computer Communications (INFOCOM), pp. 2227–2235 (2015)
17. Yan, R., Song, Y., Li, C., Zhang, M., Hu, X.: Opportunities or risks to reduce labor in crowdsourcing translation? Characterizing cost versus quality via a pagerank-hits hybrid model. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 1025–1032 (2015)
18. Yuen, M., King, I., Leung, K.: TaskRec: a task recommendation framework in crowdsourcing systems. Neural Process. Lett. **41**(2), 223–238 (2015)
19. Zhang, J., Chen, Y., Hong, S., Li, H.: REBUILD: graph embedding based method for user social role identity on mobile communication network. In: Tan, Y., Takagi, H., Shi, Y. (eds.) DMBD 2017. LNCS, vol. 10387, pp. 326–333. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61845-6_33