# Budget-feasible User Recruitment in Mobile Crowdsensing with User Mobility Prediction

Wenjie Yang[1], Guodong Sun[1,2], Xingjian Ding[3], Xiaoyue Zhang[1,2]
[1] School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China
[2] Embedded Networked Systems Lab, Beijing Forestry University, Beijing 100083, China
[3] Department of Computer Science, Renmin University of China, Beijing 100086, China
Email: {chrisyoung, sungd, zhangxiaoyue}@bjfu.edu.cn, dxj@ruc.edu.cn

***Abstract*—Mobile crowdsensing (MCS) is a new and promising tool in urban sensing. It exploits a crowd of smartphone-carried mobile users and transfers their sensory data to requesters who usually publish spatio-temporal tasks of sensing city area. In reality, mobile users can probabilistically move in the sensing region in their daily mobility and stay there for a period of time; and then these probabilistic users can be recruited to collaboratively perform MCS sensing tasks. Such an MCS depending on the probabilistic collaboration of mobile users is usually called *non-deterministic* MCS. In this paper, we focus on the budget-feasible user recruitment (BFUR) problem in non-deterministic MCS, which is the first work to maximize the requester's utility under a given budget constraint. Because of the NP-hardness of BFUR, we reformulate it as a monotone submodular maximization problem and propose a greedy algorithm (called uMax) with provable constant-factor competitiveness. Unlike previous works for non-deterministic MCS, however, this paper specially puts effort on predicting the mobility patterns of users, especially their stay time in requester's sensing region, and then designs an effective predictor based on bi-directional long short-term memory neural network. Such a prediction of user's stay time not only connects the BFUR problem modeling defined in this paper and the actual mobility uncertainty of users, but also can apply to any non-deterministic MCS campaign that depends on the knowledge of user's stay patterns. We finally validate the performance of the proposed predictor under a real-world dataset of wireless mobile networks, and evaluate algorithm uMax by comparing it with two other baseline algorithms.**

***Index Terms*—Mobile Crowdsensing, user recruitment, user mobility prediction, LSTM neural network, submodular maximization.**

## I. Introduction

Recently, most smartphones have built-in sensors that can measure motion, orientation, light strength, and other environmental conditions. The popularization of smartphones enables a new sensing paradigm, which is called mobile crowdsensing (MCS) or participatory sensing [1], [2], [3], [4]. Generally, an MCS campaign involves three types of components: the requester, the platform, and the user (also called participant). The requester publishes sensing tasks to the platform which manages to recruit smartphone users to complete these tasks; once the users recruited or chosen returns their sensory data to the platform or the requester, they will receive rewards of some type. Mobile crowdsensing is considered as a promising means of monitoring urban area [5], [6], [7].

978-1-5386-6808-5/18/$31.00 2018 IEEE

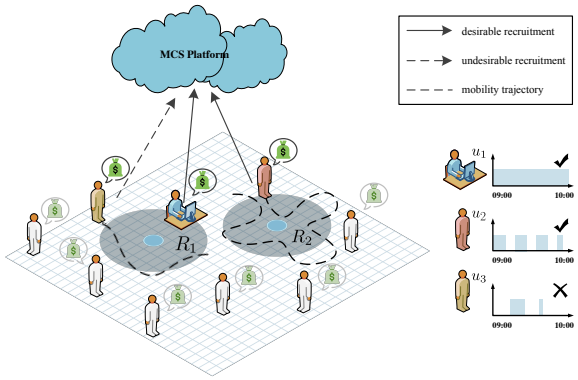



Fig. 1. An example of the user recruitment in non-deterministic mobile crowdsensing, in which gray circles represent requesters' sensing regions and dashed lines, users' trajectories.

User recruitment or user selection is an essential part of MCS application, because it dominates the service quality that the platform can provide to the requester. In this paper, we consider a more realistic MCS scenario called *non-deterministic* MCS. Unlike previous work, the non-deterministic MCS recruits probabilistic users who move into sensing regions with some probabilities because of their uncertain or irregular mobility; and consequently, multiple potential users as to a given sensing region are needed to be recruited to perform the sensing task in a collaborative way. Fig. 1 profiles the non-deterministic MCS scenario considered in our study, which involves a platform, a requester, three potential users ($u_1, u_2$, and $u_3$) registering in the platform. The requester is concerned with the signals or events occurring in two regions, $R_1$ and $R_2$, and such a region is called a *region of interest* (RoI). And then the requester publishes to the platform two timed sensing tasks ($\tau_1$ and $\tau_2$) and each task is associated with a distinct RoI; in this example, each task is to monitor the events that will come up in the associated RoI from 9:00 am to 10:00 am. Additionally, the requester requires the mobile participants to stay in RoIs for at least 30 minutes in total, aimed at acquiring sufficient spatio-temporal sensory data. Of course, the requester will pay the users as much as they charge if they are recruited to participate in MCS campaign. If the

platform knows the exact trajectories of the three users in the one-hour *task interval* (9:00 am-10:00 am), then it can determine the *stay time* of each user in given RoIs and make more desirable recruitment. In Fig. 1, we assume that within the given task interval, users $u_1$ and $u_2$ will stay in RoIs $R_1$ and $R_2$ for 60 and 35 minutes, respectively, and that user $u_3$ will stay in RoI $R_1$ for only 15 minutes. In such a case, a desirable recruitment is one that assigns users $u_1$ and $u_2$ respectively to tasks $\tau_1$ and $\tau_2$, excluding $u_3$, because $u_3$ has a low probability of moving in $R_1$ and cannot offer a stay time expected by requester.

In reality, however, it is hard for the platform to certainly know whether the user will move into requester's RoIs and whether the user's actual stay time qualifies to participate in the MCS campaign. In other words, the users' mobility and the distribution of their stay time in the task interval are both non-deterministic before such an MCS campaign starts. So, it is a crucial challenge for the platform to provide the requester with desirable service quality in an MCS campaign depending on non-deterministic users. On the other hand, in such a non-deterministic MCS, the requester naturally tends to ask a question: *how well can my tasks be performed with a given budget*, or *is my budget enough to trade off a service with the quality as expected*? To answer the question raised by requester or to systematically and effectively fulfill MCS campaign, the platform needs to dedicate to accomplishing two tasks: (1) to predict the probability that every user can move into requester's RoI in the task interval and stay there for a period of time needed by the requester, and (2) based on the obtained spatio-temporal stay characteristics of the potential users, to select proper users such that the requester's utility can be maximized under the budget constraint.

In this paper, we focus on the budget-feasible user recruitment problem (BFUR) in non-deterministic MCS scenario, and in particular, we study the accurate prediction of user's spatio-temporal stay characteristics, which provides the parameters necessary for the BFUR modeling. In the past of few years, there have been a lot of works focusing on user recruitment or task allocation in MCS. But most of them assume that user's trajectory is deterministic, that is, the platform exactly knows which users can participate in the sensing tasks issued by requester. Such an assumption is reasonable only when the sensing cycle has a long time span, such as a few weeks or even longer. On the other hand, several recent works consider non-deterministic MCS scenarios, yet they merely assume that the mobility patterns of user are known a priori, without paying attention to characterizing users' mobility patterns during MCS campaign. In fact, mining or prediction of human mobility is also a hot topic in the communities of social networks and wireless mobile networks. However, the already-existing related works usually focus either on predicting where a user is likely to move in the future, or on profiling user's spatial trajectory. These works cannot effectively yield user's spatio-temporal stay characteristics (parameters) that are highly needed to formulate the proposed BFUR problem. More specifically, the major contributions of this paper are as follows.

- To the best of our knowledge, this is the first work to investigate the BFUR problem to maximize the utility of requester with a budget constraint in non-deterministic MCS. We prove the NP-hardness of the BFUR problem.
- We reformulate the BFUR problem as a budgeted submodular maximization problem with knapsack constraint. We then propose a greedy algorithm, called uMax, which can achieve an approximation ratio of $(1 - \frac{1}{e})$ with polynomial time complexity.
- We first employ the bi-directional long short-term memory (BLSTM) neural network to predict user's spatio-temporal stay as to requester's sensing tasks. Besides yielding parameters necessary for the BFUR model, such a prediction about user mobility pattern can also serve as a key component in any MCS involving non-deterministic users.
- We use a real-world trace to evaluate the BLSTM-based approach to predicting user's stay characteristic, and we conduct extensive numeric experiments to compare uMax with two baseline algorithms. The experimental results demonstrate the significant performance of our designs.

The remainder of this paper is organized as follows. Section II defines the problem to be addressed and analyzes its challenges. Section III presents a BLSTM network model to predict the mobility of user and a greedy algorithm for the BFUR problem. Section IV involves extensive experiments to evaluate our designs. Section V introduces the works related to our study. Section VI finally concludes this paper.

## II. Model and Problem

### A. Description of Problems

We consider a general MCS scenario with non-deterministic mobile users, which is described as follows. There is a set $\mathcal{U}$ of $m$ mobile users who register at the platform and are potential participants of the MCS campaign to be scheduled by the platform. A requester publishes to the platform a set $\mathcal{T}$ of $n$ tasks which have the same starting time $t_\alpha$ and ending time $t_\beta$. Let $T = (t_\alpha - t_\beta)$ and call $T$ the *task interval*. For each task $\tau_j$, the requester specifies a region of interest (RoI), denoted by $R_j$; and specifically, task $\tau_j$ is said to be executed when some mobile user can stay in $R_j$ for a period of at least $T_{min}(T_{min} < T)$ and return his sensory data. Assuming the RoI of each task $\tau_j$ is a disk of radius $r_j$, we use a tuple $\langle R_j, r_j, t_\alpha, T, T_{min}\rangle$ to characterize the spatio-temporal property of $\tau_j$. Since the potential users may be non-deterministic in location and stay time, the requester usually expects to know, when she publishes her tasks, how well her tasks will be executed if her payment to the platform is not beyond a budget $B$.

We assume that the platform has the information about all the registered mobile users, including their historical trajectory, their current location, their cost (bid) for participating in the upcoming MCS campaign. After receiving the description of requester's tasks, the platform first sets about predicting the

TABLE I
MAIN NOTATIONS FOR THE BFUR PROBLEM

| Parameter | Description |
|---|---|
| $\mathcal{U}$ | the set of available users in the platform |
| $\mathcal{T}$ | the set of sensing tasks published to the platform |
| $\mathcal{C}$ | the set of costs of the tasks in $\mathcal{T}$ |
| $R_j$ | the region of interest of task $\tau_j \in \mathcal{T}$ |
| $r_j$ | the radius of $R_j$ |
| $T$ | the task interval |
| $T_{min}$ | the minimum stay time required by the sensing task |
| $t_\alpha$ | the start time of sensing task |
| $p_{ij}$ | the probability of user $u_i$ moving into $R_j$ after $t_\alpha$ and staying there for a period of at least $T_{min}$ |
| $\mathcal{P}$ | the set of probabilities $p_{ij}$ for $u_i \in \mathcal{U}$ and $\tau_j \in \mathcal{T}$ |
| $B$ | the budget for an MCS campaign |
| $\nu_j(\cdot)$ | the utility function for task $\tau_j$ |
| $\nu(\cdot)$ | the utility function for the requester |

users' mobility characteristics—computing $p_{ij}$, the probability that each user $u_i \in \mathcal{U}$ will move into RoI $R_j$ of task $\tau_j$ after time point $t_\alpha$ and will stay there for at least $T_{min}$. Let $\mathcal{P} = \{p_{ij}\}$ where $1 \le i \le m$ and $1 \le j \le n$. And then, with $\mathcal{P}$,$\mathcal{U}$, and $\mathcal{T}$ as input information, the platform determines a user recruitment or selection such that the tasks of the requester can be executed with as a large utility as possible while not violating the budget. In this paper, therefore, the platform needs to carry out two tasks: determining parameters $p_{ij}$ as accurately as possible and selecting a proper subset of users to maximize the utility of requester. Followed are three definitions that collectively formulate the user recruitment problem to be addressed in this paper.

**Definition 1.** *Utility of Task. For non-deterministic mobile crowdsensing, we define the utility of a task $\tau_j \in \mathcal{T}$ with the probability that it can be executed successfully in a collaborative way. Assume that a subset $U_{+j}$ of $\mathcal{U}$ is assigned to $\tau_j$ and denote the utility of $\tau_j$ by $\nu_j(U_{+j})$, which can be written as*

$$\nu_j(U_{+j}) = 1 - \prod_{u_i \in U_{+j}} (1 - p_{ij}) \tag{1}$$

Clearly, the utility $\nu_j(U_{+j})$ of task $t_j$ is an increasing set function on $2^{\mathcal{U}}$, and it ranges in (0,1].

**Definition 2.** *Utility of Requester. If a requester submits to the platform $n$ tasks ($n \ge 1$) and a subset $U_+$ of $\mathcal{U}$ is selected to execute these tasks, then we define the utility of this requester with $\nu(U_+)$ given in Eq. (2), in which $U_{+j}$ is the set of users who can execute task $\tau_j$ and set $U_+$ is the union of all non-empty $U_{+j}$.*

$$\nu(U_+) = \sum_{\tau_j \in \mathcal{T}} \nu_j(U_{+j}) \tag{2}$$

The utility of a requester, $\nu(U_+)$, can be interpreted as the total fulfillment achieved by user set $U_+$ on all the tasks of that requester. Furthermore, $\frac{\nu(U_+)}{n}$ represents the expected number of tasks that can be successfully executed in a single MCS campaign.

**Definition 3.** *Budget-Feasible User Recruitment Problem (BFUR). In a mobile crowdsensing campaign, a requester publishes a set of tasks. The platform aims to select a subset of users such that the utility of requester is as large as possible, while the total payment to the users chosen is not beyond a predefined budget.*

Formally, the BFUR problem can be formulated as the following integer programming model, which involves a non-linear objective function and a linear (knapsack) constraint function.

$$\textbf{max}: \quad \nu(U_+) \tag{3}$$

$$\textbf{st.} \quad \sum_{u_i \in U_+} c_i \le B \tag{4}$$

where $c_i$ represents the cost that user $u_i$ asks for if it is selected to perform the requester's tasks. We assume $\sum_{u_i \in \mathcal{U}} c_i > B$; otherwise, the platform can provide an optimal utility of requester, merely by recruiting all potential users, without concerns about the violation of budget $B$. The parameters as to the proposed BFUR problem are given in Tab. I.

### B. Challenges

We will first prove the NP-hardness of the BFUR problem and then demonstrate the necessity of user mobility knowledge in mobile crowdsensing user recruitment.

**Theorem 1.** *The BFUR problem is NP-Hard*

*Proof:* We prove this theorem by a straightforward reduction to the budgeted maximum coverage problem (a proven NP-hard problem). Denote by $\mathcal{T}_i$ the set of tasks that user $u_i$ can execute, and then we obtain a collection of sets $\Gamma = \{\mathcal{T}_1, \mathcal{T}_2 \cdots \mathcal{T}_m\}$, for a given BFUR problem with $m$ users. And, associate each set $\mathcal{T}_i$ with the cost of $c_i \in \mathcal{C}$. We consider a special case of the BFUR problem, where (1) $\mathcal{P} = \{p_{ij} = 1 \mid u_i \in \mathcal{U} \text{ and } \tau_j \in \mathcal{T}_i\}$ , (2) $\mathcal{C} = \{c_i = b \mid b > 0 \text{ and } u_i \in \mathcal{U}\}$, and (3) $b < B < b \times m$.

Once user $u_i$ is chosen by the platform in this case, for any task $\tau_j \in \mathcal{T}_i$, we will always have $\nu_j(\{u_i\}) = 1$. For this special case of BFUR, therefore, maximizing requester's utility is equivalent to maximizing the number of tasks that can be executed (or covered) under the constraint of budget $B$. For any task $\tau_j$, furthermore, we associate $\tau_j$ with a weight $w_j = 1$. By doing so, the special case presented here will then be reformulated as a standard budgeted maximum coverage problem [8]: it aims at maximizing the total weight of tasks (elements) covered by the subset of $\Gamma$, while the total cost is not beyond budget $B$. Since the budgeted maximum coverage problem is a well known NP-hard problem, we can conclude that the general BFUR problem is at least NP-hard. ■

The BFUR problem is computationally intractable; besides, achieving reliable user recruitment optimization in reality faces another challenge—determining which user can exactly execute the tasks published by requester. In other words, the platform has to carefully consider how to accurately estimate parameter $p_{ij}$, the probability that user $u_i$ can stay in the RoI of task $\tau_j$ for a period of time needed. Strictly, the

TABLE II
MAIN NOTATIONS FOR THE BLSTM MODEL

| Parameter | Description |
|---|---|
| $\boldsymbol{\iota}_t$ | the output vector of the input gate at timestep $t$ |
| $\boldsymbol{\phi}_t$ | the output vector of the forget gate at timestep $t$ |
| $\boldsymbol{\omega}_t$ | the output vector of the output gate at timestep $t$ |
| $\boldsymbol{\theta}_t$ | the output vector of the internal state at timestep $t$ |
| $\boldsymbol{h}_t$ | the output vector of the hidden layer at timestep $t$ |
| $\boldsymbol{y}_t$ | the output vector of BLSTM at timestep $t$ |
| $W$ | the weight matrices of the corresponding units |
| $\boldsymbol{s}$ | the sequence of user $u$'s historical stay time at $\langle R, r\rangle$ |
| $\boldsymbol{x}_t$ | the input vector at timestep $t$, used as input data for BLSTM |
| $\boldsymbol{z}_t$ | the label vector at timestep $t$, used as input data for BLSTM |
| $K$ | the number of classes (labels) that equally segment the task interval $T$ |
| $C_k$ | the $k$-th class representing that the stay time of user in some RoI ranges in $\left[\frac{k-1}{K}T, \frac{k}{K}T\right)$ |
| $S$ | the training dataset |
| $X$ | the input data (containing the data needed both by training and by prediction) |
| $l$ | the look-back windows size of the input vector |
| $\mathcal{L}$ | the Cross-Entropy loss function |

platform cannot entirely make sure whether a user will move to or stay in a given RoI. Therefore, most works on user recruitment assume that parameter $p_{ij}$ is exactly known in advance, skipping over its determination. In practice, however, the prediction of user's stay time is more challenging than that of user's future location. Until now, there is still lack of effective approaches.

## III. DESIGNS

Before presenting a greedy algorithm for the BFUR problem, we first make prediction of the user mobility—to obtain the probabilistic knowledge of user's future mobility characteristics—which are necessary parameters for modelling the BFUR problem.

### A. BLSTM-based Prediction of User Mobility

The recurrent neural network (RNN) approaches [9], [10] have limitations on long-term dependency and context generalization; such limitations unavoidably degrade the performance of prediction, and consequently, cannot meet the requirement of mobile crowdsensing for effective user recruitment. In this paper, we propose a novel method that predicts the user mobility characteristics with a bidirectional long short-term memory (BLSTM) neural network model. Tab. II describes the major notations for the BLSTM model.

*1) Basics of BLSTM:* Given an input sequence $\boldsymbol{x}$, the standard RNN model recursively computes the sequences of hidden vectors $\boldsymbol{h}$ and outputs vectors $\boldsymbol{y}$. The BLSTM model is a variant of RNN, and it is essentially a network that is composed of a special type of neuron (also known as memory block). Each block contains one or more recurrently-connected memory cells. Each cell involves three non-linear units: the input gate $\iota$, the forget gate $\phi$, and the output gate $\omega$. Additionally, each cell provides an internal state (memory) $\theta$ to each gate. These gates collect activation from both inside and outside the memory block and control cell's activation. Specifically, the surrounding parts of a memory cell can interact with it only through its gates. The equations from (5) to (9) define how the states in an LSTM layer of memory cells is updated at every timestep $t$, where the $W$ terms represent the weight matrices of the corresponding units [11].

$$\boldsymbol{\iota}_t = \sigma(W_{x_\iota}\boldsymbol{x}_t + W_{h_\iota}\boldsymbol{h}_{t-1} + W_{\theta_\iota}\boldsymbol{\theta}_{t-1}) \quad (5)$$

$$\boldsymbol{\phi}_t = \sigma(W_{x_\phi}\boldsymbol{x}_t + W_{h_\phi}\boldsymbol{h}_{t-1} + W_{\theta_\phi}\boldsymbol{\theta}_{t-1}) \quad (6)$$

$$\boldsymbol{\theta}_t = \boldsymbol{\phi}_t\boldsymbol{\theta}_{t-1} + \boldsymbol{\iota}_t \tanh(W_{x_\theta}\boldsymbol{x}_t + W_{h_\theta}\boldsymbol{h}_{t-1}) \quad (7)$$

$$\boldsymbol{\omega}_t = \sigma(W_{x_\omega}\boldsymbol{x}_t + W_{h_\omega}\boldsymbol{h}_{t-1} + W_{\theta_\omega}\boldsymbol{\theta}_t) \quad (8)$$

$$\boldsymbol{h}_t = \boldsymbol{\omega}_t \tanh(\boldsymbol{\theta}_t) \quad (9)$$

Specifically, the hidden layer is split into two separate layers in forward and backward directions. The positive time direction data and the negative time direction data are fed to the corresponding layers, and then after being processed, both can be concatenated in the same output layer by using Eq. (10), where the forward and the backward hidden layer output sequences $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ are normalized by the softmax activation function [12], ensuring that all the elements of the network output vector $\boldsymbol{y}_t$ are between 0 and 1, and that they sum up to 1 on every timestep.

$$\boldsymbol{y}_t = \text{softmax}(W_{\overrightarrow{\boldsymbol{h}}y}\overrightarrow{\boldsymbol{h}}_t + W_{\overleftarrow{\boldsymbol{h}}y}\overleftarrow{\boldsymbol{h}}_t) \quad (10)$$

*2) Prediction of User's Stay Characteristics:* Before an MCS campaign can be launched, the requester usually specifies her tasks and their spatio-temporal requirement as well as total budget. Thus, the platform needs to know which users will move into requester's RoIs and stay there for a period of time at least $T_{min}$; in other words, it needs to predict the user's stay characteristics that will be actually reflected after the tasks are published.

In this paper, we employ the BLSTM model to predict user's stay characteristics by answering the following question: *how likely will a user stay in a given RoI for a period of time at least* $T_{min}$? In other words, for any given task $\tau_j$ and user $u_i$, the proposed BLSTM model outputs the parameter $p_{ij}$ needed by the platform to model and solve the BFUR problem. For brevity, we omit the subscripts $i$ and $j$ representing the IDs of user and RoI. Given a task $\tau$, we denote by $\boldsymbol{s}$ the sequence of user $u$'s historical stay time at $\langle R, r\rangle$. First, the platform reprocesses $\boldsymbol{s}$ and $\tau$: assembling the first several timesteps of $\boldsymbol{s}$ into the input vector $\boldsymbol{x}$, normalizing $\boldsymbol{x}$ to the range of $[-1, 1]$, and encoding the sum of next few timesteps of $\boldsymbol{s}$ into a label vector $\boldsymbol{z}$. Second, the platform keeps iterating to the next timestep—repeating the previous steps—until all the available data as to $\boldsymbol{s}$ and $\tau$ is included. After these iterations are completed, the platform yields the training dataset $S$. Third, with $S$ as input, the BLSTM can be trained and then the optimal parameters can be achieved. Finally, at a proper moment before $t_\alpha$, the trained BLSTM receives a prior input and makes a prediction of $\boldsymbol{y} = (y^k)$, where $y^k$ represents the estimated conditional posterior probability of class $k$. Recall that each class represents the duration of user's stay. With

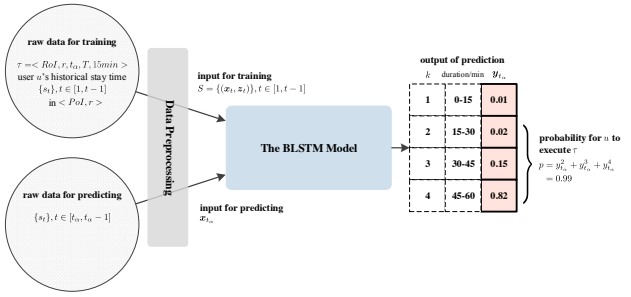



Fig. 2. Example of the BLSTM model that predicts user $u$'s stay characteristics at the RoI of task $\tau$, in which task interval $T$, $K$, and $T_{min}$ are set to one hour, 4, and 15 minutes, respectively.

doing so, therefore, we can deduce how likely the duration that $u$ will stay at $\langle R, r\rangle$ belongs to some class. According to these classification results and the corresponding probabilities in $\boldsymbol{y}$, the platform can then compute $p$ with the specified $T_{min}$ of $\tau$.

Next we take a concrete example shown in Fig. 2 to tangibly demonstrate the steps of the proposed BLSTM model to complete training and prediction. In real-world applications, different tasks may have different demands on $T_{min}$. Thus, the user mobility prediction cannot simply be modeled as a binary classification problem. To address this practical issue, for a single task interval starting at $t_\alpha$, we set $K(K > 2)$ classes (also called labels) as to the ranges of the possible total stay time. We use term $C_k(1 \le k \le K)$ to denote the $k$-th class and in detail, $C_k$ represents the probabilistic event that the total stay time of user ranges in time interval $\left[\frac{k-1}{K}T, \frac{k}{K}T\right)$. If $T = 1$ hour, $K = 4$, and $\mathbf{Pr}(C_2) = 0.23$, for example, it is indicated that the stay time of user (in minute) will be within $[15, 30)$ with probability of 0.23. Furthermore, our prediction of user mobility qualifies as a 1-of-$K$ classification problem on a time-series dataset, and it can return the vector $\boldsymbol{y}_{t_\alpha}$. And, $y^k_{t_\alpha}$, the $k$-th element of $\boldsymbol{y}_{t_\alpha}$, measures the probability that the stay time of user can be categorized as class $C_k$. For any given $T_{min}$, therefore, we can predict how likely the total stay time of user will be longer than $T_{min}$, by accumulatively considering the distribution of $y^k_{t_\alpha}$.

For such a 1-of-$K$ classification problem, the first step for the BLSTM model is to assemble user $u$'s training dataset $S$, which is a sequence of input-label pairs $(X, Z)$. More specifically, at timestep $t$, the input is a size-fixed vector $\boldsymbol{x}_t = (s_t)$ with $t \in [t-l, t-1]$, where $l$ is the look-back window size; the label $\boldsymbol{z}_t$, a $k$-dimensional vector, is equal to $q(\sum_{t=t}^{t+T} s_t)$, where function $q(\cdot)$ encodes the sum of $u$'s stay times from $t$ to $(t+T)$ according to which class it belongs to. If $t = 1, T = 1, K = 4$, and $s_1 = s_2 = \frac{T}{4}$, for instance, $q(\sum_{t=1}^{2} s_t) = (0, 1, 0, 0)$. After the above preprocessing, we can apply a standard supervised learning policy to train the BLSTM model, aimed at learning the dependency between the historical data and the future data. In detail, we employ the Adam optimization algorithm, an extension of the stochastic gradient descent [13], to determine the weight matrices (termed $W$) for the input, hidden and output layers given in Eq.(5)-(9). Such a training process of Adam minimizes the Cross-Entropy loss function that measures the difference between the estimated and the actual stay characteristics of user. The Cross-Entropy loss function in use is given in Eq.(11)

$$\mathcal{L}(S) = -\sum_{\boldsymbol{z}_t, \boldsymbol{y}_t \in S} \boldsymbol{z}_t \log \boldsymbol{y}_t \tag{11}$$

As explained in [14], after the softmax-based normalization in Eq. (10), the relative magnitude of the network output $\boldsymbol{y}_t$ can be interpreted as an estimate of the posterior probability of being some class conditioned on all the inputs $X$:

$$\begin{aligned} y^k_t &= \mathbf{Pr}(C_k|X) \\ &= \mathbf{Pr}\left(\frac{k-1}{K}T < \sum_{t=t}^{t+T} s_t \le \frac{k}{K}T \,\middle|\, X\right), k \in [1, K] \end{aligned} \tag{12}$$

Then the conditional probability that the user's stay time is greater than or equal to $T_{min}$ can be evaluated with (13).

$$p = \mathbf{Pr}\left(\sum_{t=t_\alpha}^{t_\alpha+T} s_t \ge T_{min} \,\middle|\, X\right) \ge \sum_{k=k'}^{K} y^k_{t_\alpha} \tag{13}$$

where $k'$ is set to $\lceil \frac{T_{min}}{T/K} \rceil + 1$, the sequence of the first sub-interval whose length is not less than $T_{min}$. In the case with integer $\frac{T_{min}}{T/K}$, we have $p = \sum_{k=k'}^{K} y^k_{t_\alpha}$.

The example of Fig. 2 shows the computation of $p$. With $x_{t_\alpha}$ as input, the BLSTM model outputs the estimation results $y_{t_\alpha}$. As $K = 4$ and $T = 60$ min, $y_{t_\alpha}$ is a 4-dimensional vector and each element of it indicates the probability that $u$'s stay time belongs to a sub-interval with 15 min duration. Here, the resulted $\boldsymbol{y}_{t_\alpha}$ is $(0.01, 0.02, 0.15, 0.82)$ and $k' = \frac{T_{min}}{T/K} + 1 = 2$. By summing the probabilities from classes 2 up to 4 in $y_{t_\alpha}$, therefore, we have $p = 0.99$. The experimental results in IV-A show that in most cases, the proposed BLSTM model can achieve more-than-90% accuracy in predicting the stay time characteristics of user.

### *B. Algorithm for the BFUR problem*

Since the BFUR problem is NP-hard, we design a greedy algorithm, called uMax, aimed at obtaining an approximate solution for maximizing requester utility. Described in Algorithm 1, the proposed uMax consists of two successive phases.

In the first phase, by enumeration, uMax finds out a partial solution that involves only three users such that the requester utility is as large as possible. Such a partial enumeration is a widely-adopted method that helps the greedy algorithm achieve a constant approximation ratio by sacrificing a bit of computation time [15], [16], [17]. This phase enumerates out $\mathcal{O}(m^3)$ three-user subsets of user set $\mathcal{U}$, where $m$ is the size of $\mathcal{U}$. For any three-user subset, we need to compute the utility of each task in order to obtain the requester utility. According to Definition 1 and Definition 2, therefore, $\mathcal{O}(n)$ time is needed to compute the requester utility provided by any three-user

subset, where $n$ is the size of task set $\mathcal{T}$. Thus, the first phase of uMax takes $\mathcal{O}(nm^3)$ time. At the end of this phase, the set $U_+^k$ with $k = 3$ can be determined in line 1 of Algorithm 1.

In the second phase, uMax employs a while loop to greedily select users from set $U^k \backslash U_+^k$ until all the users of $U^k$ are examined. In each iteration, uMax prefers the user of $U^k \backslash U_+^k$ who can bring the largest marginal utility with unit cost. As shown in line 4, $\nu(U_+^k)$ represents the current utility of requester, and $\nu(U_+^k \cup \{u_k\})$, the expected utility of requester if user $u_k$ would be included into $U_+^k$. Thus, the difference between $\nu(U_+^k \cup \{u_k\})$ and $\nu(U_+^k)$ measures the marginal utility brought by the inclusion of $u_k$. After picking out the best candidate user $u_k$ in line 4, uMax will decide whether to add $u_k$ into $U_+^i$. If the remaining budget is enough for uMax to include $u_k$, user $u_k$ will then be moved from $U^k$ to $U_+^k$.

**Algorithm 1:** the uMax algorithm

**Input:** $\mathcal{U}$, $\mathcal{T}$, $\mathcal{P}$, and $\mathcal{C}$
**Output:** $\nu$ and $U_+^k$ (a subset of $\mathcal{U}$)

1 Enumerate all possible three-user subsets of $\mathcal{U}$ and determine $U_+^k = \{u_a, u_b, u_c\}$ such that $\nu(U_+^k)$ is maximized
2 $k = 3$ and $U^k \leftarrow \mathcal{U}$
3 **while** $U^k \backslash U_+^k \neq \emptyset$ **do**
4  Select a user $u_k \in U^k$ such that $\frac{\nu(U_+^k \cup \{u_k\}) - \nu(U_+^k)}{c_k}$ is as large as possible (break tie arbitrarily)
5  **if** $\sum_{u_i \in U_+^k \cup \{u_k\}} c_i \leq B$ **then**
6  $U_+^{k+1} \leftarrow U_+^k \cup \{u_k\}$
7  **end**
8  $U^{k+1} \leftarrow U^k \backslash \{u_k\}$
9  $k \leftarrow k + 1$
10 **end**
11 **return** $U_+^k$ and the corresponding $\nu$

### C. Performance Analysis of uMax

**Theorem 2.** *Algorithm* uMax *terminates with the time complexity of* $\mathcal{O}(nm^3)$.

*Proof:* As analyzed above, the first phase takes $\mathcal{O}(nm^3)$ time, and so, we only need to analyze the time complexity of the second phase. At the beginning of the second phase, the set $U^k$ is of size $(m-3)$, and consequently, the while loop involves $\mathcal{O}(m)$ iterations. In the $k$-th iteration, uMax finds out the "best" user $u_k$, according to the greedy criterion, and checks whether or not the budget constraint will be satisfied if $u_k$ is added into $U_+^k$. There are $(m-k)$ users to be examined one by one in the $k$-th iteration. After the best user $u_k$ is determined, the requester's utility achieved by $U_+^k \cup \{u_k\}$ can be computed in $\mathcal{O}(nm)$ time. Additionally, it takes linear time to check the constraint feasibility in each iteration, and then, the while loop terminates with the time of $\mathcal{O}(nm^2)$. Combining the two phases, we know that the overall time complexity of algorithm uMax is $\mathcal{O}(nm^3 + nm^2) = \mathcal{O}(nm^3)$. ■

Submodularity is defined as a discrete analog of convexity in continuous optimization. The properties of submodular function play important role in the combinatorial optimization and the design of greedy algorithm. Next we prove the submodularity implied in our problem by Lemma 1.

**Lemma 1.** *The requester utility function* $\nu(U)$ *is submodular for any* $U \subseteq \mathcal{U}$.

*Proof:* By the definition of the requester utility function shown in Eq. (2), we have (1) $\nu(U)$ formally is a linear combination of a set of task utility functions, (2) $\nu(\emptyset) = 0$, and (3) $\nu(U) > 0$ for any $U \neq \emptyset$ and $U \subseteq \mathcal{U}$.

Thus we only need to prove the task utility function $\nu_j(U)$ is submodular for any given task $\tau_j$. Clearly, the three properties listed above are also true for function $\nu_j(U)$. Consider two subsets $U_1$ and $U_2$ such that $U_1 \subset U_2 \subset \mathcal{U}$. For any task $\tau_j$ and a user $u_k \in \mathcal{U}$, we denote by $\Delta_{u_k}\nu_j(U_1)$ the evaluation of $(\nu_j(U_1 \cup \{u_k\}) - \nu_j(U_1))$. Clearly, we have $\Delta_{u_k}\nu_j(U_1) = 0$ for $u_k \in U_1$. In the case where $u_k \in \mathcal{U} \backslash U_1$, we have

$$\begin{aligned}\Delta_{u_k}\nu_j(U_1) &= 1 - \prod_{u_i \in U_1 \cup \{u_k\}} (1 - p_{ij}) - 1 + \prod_{u_i \in U_1} (1 - p_{ij}) \\ &= p_{kj} \times \prod_{u_i \in U_1} (1 - p_{ij})\end{aligned}$$

Similarly, we obtain $\Delta_{u_k}\nu_j(U_2) = p_{kj} \times \prod_{u_i \in U_2} (1 - p_{ij})$. If $u_k \in U_2$, we have $\Delta_{u_k}\nu_j(U_2) = 0$. It is clear to know that the function of form $\prod_{u_i \in \mathcal{U}}(1 - p_{ij})$ over domain $\mathcal{U}$ is monotone decreasing for any given task $\tau_j$. Hence we have $\Delta_{u_k}\nu_j(U_1) \geq \Delta_{u_k}\nu_j(U_2)$, which furthermore indicates

$$\nu_j(U_1 \cup \{u_k\}) - \nu_j(U_1) \geq \nu_j(U_2 \cup \{u_k\}) - \nu_j(U_2)$$

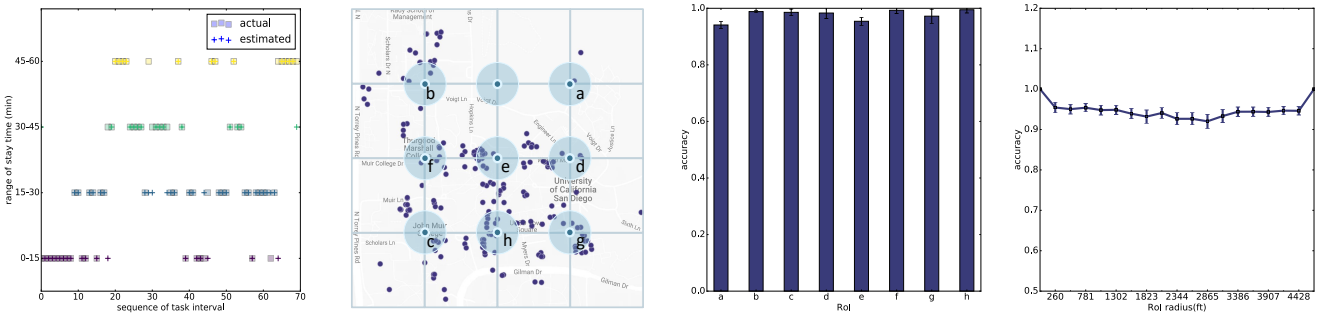
for any $U_1 \subset U_2 \subset \mathcal{U}$ and any $u_k \in \mathcal{U}$. Function $\nu_j(U)$ is then submodular over the domain of $2^{\mathcal{U}}$. ■

Considering the above lemma and the optimization objective of the BFUR problem, we easily conclude that this problem is basically a budgeted submodular maximization problem with knapsack constraint (BSM-KC), which is a special case of the budgeted maximum coverage problem [15], [18], [16] and proves NP-hard in [8]. A greedy algorithm [15] was presented to approximately maximize a non-decreasing submodular set function with budget constraint, and its performance guarantee is $(1 - \frac{1}{e})$ in the worst case. Inspiring our work, this algorithm guesses the first three items of the approximate solution, and then it determines the other items, filling up the remaining budget. In [15], it is proven that there exists a greedy algorithm for the general submodular coverage maximization with a knapsack constraint, which can achieve a competitive ratio of $(1 - \frac{1}{e})$. Since algorithm uMax employs a greedy criterion similar with [15], it is easy to prove that the approximation ratio of uMax is also $(1 - \frac{1}{e})$. We omit the proof details due to the page limitation.

## IV. Evaluation

### A. Performance of User Mobility Prediction

*1) Dataset and Settings:* To evaluate the prediction performance of the presented BLSTM model, we use the *Wireless*




(a) prediction of a user's stay time (b) RoIs of different AP densities (c) RoI vs. accuracy (d) RoI radius vs. accuracy

Fig. 3. Performance of the BLSTM-based user mobility prediction

*Topology Discovery* (WTD) dataset of mobile users [19], which has also been used in [20]. The WTD dataset involves the 11-week communication logs of 275 PDAs (users) and over 500 APs deployed in the campus of University of California, San Diego. After excluding a few anonymous APs, we obtained totally 13,215,413 communication logs between 209 PDAs and 197 APs that was recorded from September 7, 2001, to December 7, 2001. Since most of these APs are location-fixed and their locations are known a priori, we can treat a RoI to be a disk that covers a set of APs; specifically, if a user can keep connection with at least one of APs in some RoI, we say that this user locates in this RoI.

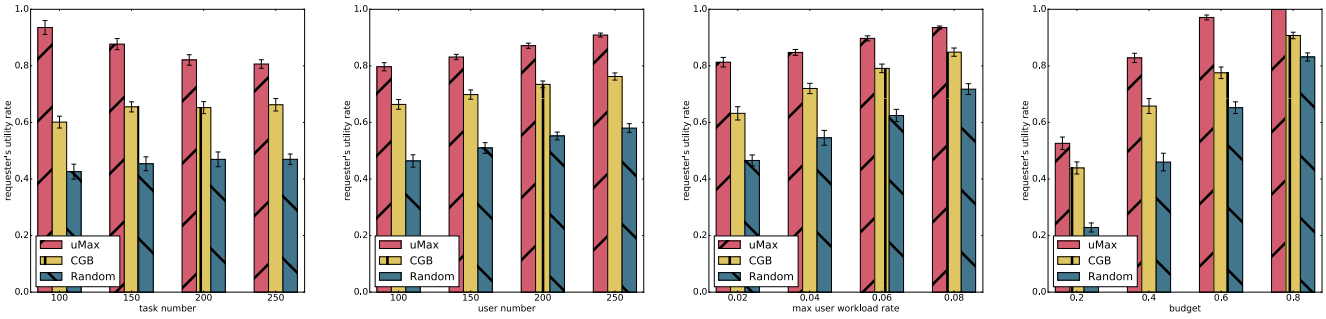
The implementation of our BLSTM model contains three hidden layers, each including forward and backward layers. The three layers involve 576, 144, and 72 one-cell memory blocks, respectively, and all the $W$ matrices involve 628,132 weights in total. The input and the output layer are both four in size, and both of them are fully connected with the hidden layers. Functions tanh and softmax are used in the BLSTM model to serve as the activation functions for the gates and the output layer, respectively. In experiments, 80% of user's historical data was used as the training set, and the remaining part, as the testing set. The time span of two successive timesteps was set to one hour, and the task interval $T$, also to one hour. The size of the look-back window, $l$ and the number of classes, $K$, were both set to four.

*2) Results:* To directly demonstrate the performance of our BLSTM model, we picked out a user whose ID is 164 in the WTD dataset and whose mobility is irregular in comparison with others' mobility. With the proposed BLSTM mode and a given RoI, we predicted the stay characteristics of this user in each of 70 successive hours and the results are plotted in Fig. 3(a); in other words, with task interval $T$ set to one hour, we made 70 repetitive prediction experiments for the selected user, each with incremental starting times. It can be seen in Fig. 3(a) that almost all the predicted ranges (classes) of stay time are consistent with the actual ones.

To comprehensively evaluate our predictor, we set eight radius-identical RoIs, labeled $a$, $b$, $c$...up to $h$, which are shown in Fig. 3(b) and have successively greater AP densities. Here we measure the AP density of a RoI with the number of APs locating in this RoI divided by the total number of APs in the whole campus. With the task interval set to a single hour, we examined the stay characteristics of all mobile users with respect to each RoI of Fig. 3(b), and the average prediction results are shown in Fig. 3(c) with a confidence of 0.95. We can find that the average prediction accuracy achieved for each RoI is higher than 94%, and that the average accuracies for RoIs $b$ and $h$ almost reach 100%. Specially, Fig. 3(c) also shows that the AP density of a RoI puts an irregular but insignificant impact on the prediction performance.

Fig. 3(d) plots the variation of prediction accuracy for RoIs with different radii. Different from the RoIs specified in Fig. 3(b), in the experiments for Fig. 3(d), we set 19 RoIs which are concentric but have evenly incremental radii ranging from 0 to 4,688 ft. The RoIs with radii of 0 ft and 4,688 ft are two extreme cases, both of which did not cover any AP and covered all the APs of the WTD dataset, respectively. Fig. 3(c) and Fig. 3(d) collectively demonstrate the robustness of the proposed BLSTM-based predictor. Interestingly, it can be seen in Fig. 3(d) that the prediction accuracy is relatively low for the radius-medium RoIs. For the RoI of radius 2865 ft, for instance, the average accuracy is 92% and lower than those achieved for other RoIs with greater radius.

Besides the overall average accuracy shown in Fig. 3(d). Similar with the observation in Fig. 3(d), the proposed predictor seemingly works a little better for the RoIs with either small or large radius than it does for the RoIs with medium radius. The reasons behind are as follows. With radius-small RoIs, the user's mobility can be abstracted with simple and explicit patterns—some users tend to stay in such RoIs for a long time, while others often just pass through them without any transient stay at all. With radius-large RoIs, more and more users are thought of being kept in such RoIs, even though they could move all the time without any stop. For the above two kinds of cases, it is easier for our predictor to make decisions because of simple and explicit mobility patterns. For the radius-medium RoIs, however, there are possibly more complicated mobility patterns, which can neither be "pruned out" as done in radius-small RoIs nor be "hidden" as done in radius-large RoIs.




(a) Task number vs. Utility (b) User number vs. Utility (c) Max workload rate vs. Utility (d) Budget vs. Utility

Fig. 4. Comparison of requester's utility achieved by the three algorithms with different settings

### B. User Recruitment Algorithms in Comparison

For any user $u_i \in \mathcal{U}$ and any task $\tau_j \in \mathcal{T}$ in an instance of the BFUR problem, the proposed BLSTM-based predictor can employ users' historical trace data to determine parameter $p_{ij}$ and then $\mathcal{P}$, which is needed by algorithm uMax. In this subsection we evaluate uMax with extensive numeric experiments and compare it with two baseline algorithms.

*1) Experimental Methodology:* One baseline algorithm is called Random, which continues to randomly select a user from $\mathcal{U}$ until the budget is violated. The other baseline algorithm is called CGB (cost-greedy baseline), which is designed with an intuitive greedy criterion. In each iteration, CGB always selects the user with minimum cost, as long as the updated total cost is inside of budget $B$. To achieve a fair comparison, we always input the same parameters ($\mathcal{U}$, $\mathcal{T}$, $\mathcal{C}$, and $\mathcal{P}$) to the three algorithms, for a given experimental case. In experiments, we evaluated uMax and two baselines in two major aspects.

- *Requester's utility rate*. Trivially, we can maximize the utility of requester by recruiting all the users, if we need not consider the budget constraint. Let $\nu_{max}$ be the maximum requester's utility without considering the budget constraint, and let $\nu$ be the utility of requester yielded by uMax. We then define the requester's utility rate of uMax with $\frac{\nu}{\nu_{max}}$. Similar definition also suits for baseline algorithms Random and CGB.
- *Budget utilization rate*. It is defined by the ratio between the total cost of selected users and the budget $B$. Generally, a well-performed algorithm will always struggle to make full use of the budget for maximizing the requester's utility.

In experiments, we considered six cases with different parameter settings, all of which are listed in Tab. III. We repeated the experiment for each case 20 times with different starting times of task interval. The results are plotted in Fig. 4 with 0.95 confidence.

In Tab. III, the *Max workload rate* puts a limit on the maximum number of tasks that a user is allowed to execute in a single MCS campaign. For instance, if it is set to 0.02 and there are 200 tasks, then a user can execute at most four

TABLE III
CONFIGURATIONS IN EXPERIMENTS

| Case | #user | #task | range of user cost | *Budget* | *Max workload rate* |
|---|---|---|---|---|---|
| 1 | 100 | 100∼250 | [1, 8] | 0.4 | 0.02 |
| 2 | 100∼250 | 250 | [1, 8] | 0.4 | 0.02 |
| 3 | 100 | 200 | [1, 8] | 0.4 | 0.02∼0.08 |
| 4 | 100 | 200 | [1, 8] | 0.2∼0.8 | 0.02 |
| 5 | 100∼250 | 250 | [1, 8] | 0.2 | 0.02 |
| 6 | 250 | 100∼250 | [1, 8] | 0.2 | 0.02 |

(200×0.02) tasks. In each experiment, the workload rate of every user is randomly determined from zero to the *Max workload rate*, and also, the cost of every user is randomly determined within [1,8]. The *Budget* in Tab. III indicates a way of setting budget $B$. If it is set to 0.4 and the cost of all users is totally $C$, then $B$ is set to $0.4 \times C$ in experiments.

*2) Results:* Fig. 4(a) plots the effect of the number of tasks on requester's utility in Case 1. Clearly, Random performs worst. The requester's utility achieved by the proposed uMax is at least 17.8% higher than that by CGB. Additionally, the performance of uMax degrades slightly as the number of tasks increases from 100 to 250. Fig. 4(b) shows that for each of the three algorithms in Case 2, more users can result in higher requester's utility. Fig. 4(c) shows the experimental results of Case 3. It can be seen that with the increase of available users, the three algorithms all improve their performance. In repetitive experiments of this case, however, uMax can keep stable, while two other algorithms experience a little fluctuation. Fig. 4(a) and 4(c) collectively reveals that when the number of available users or the maximum workload rate per user is larger, the platform's capability of executing tasks will be stronger, leaving the performance gap between uMax and the other two algorithms slightly reduced. By Case 4, we examined the variation of requester's utility against budget $B$; and Fig. 4(d) compares the three algorithms.

## V. RELATED WORK

In human mobility-based applications, mining, profiling, or predicting human mobility characteristics helps to well utilize the potential value of crowd mobility. A lot of researches have

proposed human mobility models based on human trajectories [21], [22], [23], mobile phone records (call detail records, CDRs) [24], [25], and various paradigms of networks (such as social networks and mobile network) [26], [20], [19]. In our MCS scenario, however, the prediction target of user's mobility is rather different from that of these previous works; they usually estimate the future locations of mobile users, without giving further information about how long these users will stay in the specified region. Next we briefly introduce the prior approaches to recruiting users or allocating tasks in MCS applications.

### A. User Recruitment in Deterministic MCS

As a crucial challenge in MCS, user recruitment problem has been extensively explored by many researches. Most of these works concentrate on deterministic MCS campaign with an objective that maximizes the system utility or minimizes the application cost. In deterministic MCS, the platform exactly knows which users can participate in an MCS campaign.

Marjanovi et al. in [27] present an energy-aware MCS framework with the basic requirement of sensing quality. They first qualify the value of each sensor by an individual-based evaluation function, and then select a predefined number of the most valuable sensors. Hu et al. in [28] consider a more flexible user recruitment scenario where tasks can be executed collaboratively and users are paid according to their actual sensing load. Yang et al. [29] emphasize the timeliness of tasks in MCS and formulate a task allocation problem that aims at minimizing the penalty caused by the delay of users in collecting sensing data. They then propose a hybrid genetic algorithm and a hybrid greedy algorithm to solve their problem. Wang et al. [30] study a multi-sensor assignment problem in an energy-efficient MCS paradigm, in order to minimize the sensing energy consumption while guaranteeing the sensing quality. For the energy-efficient MCS case, Zhao et al [31] investigate a fair user recruitment framework, which minimizes the maximum sensing time of each participating user. They present online and offline algorithms for their problem. Yi et al. [32] present a fast user recruitment algorithm to maximize the utility of requesters, yet without explicit cost constraint. They also prove the linear-time complexity and the performance guarantee of the algorithm.

### B. User Recruitment in Non-deterministic MCS

In non-deterministic MCS applications, the platform often recruits multiple users and lets them collaboratively work on each common sensing tasks.

In [33], Zhang et al consider a special non-deterministic MCS scenario that leverages piggyback in collecting cellphone users' data. They profile and predict users' phone call patterns by inhomogeneous Poisson processes in each cell tower, then based on the predicting probabilities, they iteratively select a candidate who has the maximum utility and stops the recruitment once the sensing coverage probability reaches a given threshold. Similar scenario and problems have also been studied in other works [34], [35], [36], [37], [38]. Li et al. [34] study the task assignment problem and present both offline and online algorithms. In their work, however, the sensing cycle is set to a couple of weeks and the call probability is roughly defined as the ratio of call times observed. Therefore the proposed designs only suit coarse-grained sensing tasks that continue for tens of days or even a few months. Adopting the same non-deterministic MCS model as ours, Xiao et al. [39] consider an MCS scenario with probabilistic collaboration and investigate a user recruitment algorithm with deadline constraint.

These previous works considering non-deterministic MCS assume that the probabilistic user participation or the probabilistic collaboration has been evaluated by some means a priori, and do not pay much attention on how to obtain such probabilistic parameters needed by the user recruitment, thereby impacting the systematic practicability of their designs.

## VI. Conclusion

In this paper, we have studied the user recruitment problem (BFUR) with a budget constraint in the MCS with non-deterministic mobile users. The proposed BFUR problem is solved by the greedy polynomial algorithm uMax with provable approximation ratio. In particular, to draw a connection between the BFUR modeling and the uncertainty of mobile users, we have designed a BLSTM network model, by which we can accurately predict the mobility patterns of users, especially their stay time characteristics with respect to given RoIs. The presented approach to predicting the stay time of mobile users should be quite broadly applicable in non-deterministic MCS's. We have also conducted extensive experiments to evaluate the performances of the prediction approach and to compare the greedy algorithm with two baselines. The experimental results demonstrate that under a real-world dataset, the proposed BLSTM-based method can achieve more-than-90% accuracy, and that algorithm uMax outperforms the two baselines in terms of requester's utility maximization.

## Acknowledgment

This work was supported by the Fundamental Research Funds for the Central Universities of China with Grant No. 2017ZY20, and in part, by the NSF of China with Grant No. 61300180.

## References

[1] A. Antonic, M. Marjanovic, K. Pripuzic, and I. Zarko, "A mobile crowd sensing ecosystem enabled by cupus: Cloud-based publish/subscribe middleware for the internet of things," *Future Generation Computer Systems*, vol. 56, pp. 607–622, 2016.

[2] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang, "Fliermeet: a mobile crowdsensing system for cross-space public information sharing," *IEEE Transactions on Mobile Computing*, vol. 14, pp. 2020–2033, 2015.

[3] X. Hu, T. Chu, H. Chan, and V. Leung, "Vita: A crowdsensing-oriented mobile cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 147–165, 2013.

[4] J. Wang, Y. Wang, D. Zhang, F. Wang, H. Xiong, C. Chen, Q. Lv, and Z. Qiu, "Multi-task allocation in mobile crowd sensing with individual task quality assurance," *IEEE Transactions on Mobile Computing*, 2018.

[5] M. Elhamshary, M. Youssef, A. Uchiyama, H. Yamaguchi, and T. Higshino, "Transitlabel: A crowd-sensing system for automatic labeling of transit stations semantics," in *MobiSys*, Singapore, June 2016, pp. 193–206.

[6] X. Guo, E. Chan, C. Liu, K. Wu, S. Liu, and L. Ni, "Shopprofiler: Profiling shops with crowdsourcing data," in *Infocom*, 2014, pp. 1240–1248.

[7] H. Ji, L. Xie, C. Wang, Y. Yin, and S. Lu, "Crowdsensing: A crowdsourcing based indoor navigation using rfid-based delay tolerant network," *Journal of Network and Computer Applications*, vol. 52, pp. 79–89, 2015.

[8] S. Khuller, A. Moss, and J. Naor, "The budgeted maximum coverage problem," *Information Processing Letter*, vol. 70, pp. 39–45, 1999.

[9] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, 2003.

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[11] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[12] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1996.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*. Springer, 1990, pp. 227–236.

[15] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, pp. 41–43, 2004.

[16] R. Iyer and J. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *NIPS*, Lake Tahoe, Nevada, USA, 2013.

[17] J. Yu and S. Ahmed, "Maximizing expected utility over a knapsack constraint," *Operations Research Letters*, vol. 44, no. 2, pp. 180–185, 2016.

[18] J. Lee, V. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *STOC*, Bethesda, Maryland, USA, 2009.

[19] M. McNett and G. M. Voelker, "Access and mobility of wireless pda users," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 2, pp. 40–55, 2005.

[20] R.-Y. Yu, X.-Y. Xia, J. Li, Z. Yan, and X.-W. Wang, "Social-aware mobile user location prediction algorithm in participatory sensing systems," vol. 38, pp. 374–385, 02 2015.

[21] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti, "Unveiling the complexity of human mobility by querying and mining massive trajectory data," *The VLDB JournalThe International Journal on Very Large Data Bases*, vol. 20, no. 5, pp. 695–719, 2011.

[22] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the levy-walk nature of human mobility," *IEEE/ACM transactions on networking (TON)*, vol. 19, no. 3, pp. 630–643, 2011.

[23] T. Jia, B. Jiang, K. Carling, M. Bolin, and Y. Ban, "An empirical study on human mobility and its agent-based modeling," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2012, no. 11, p. P11024, 2012.

[24] H. Kanasugi, Y. Sekimoto, M. Kurokawa, T. Watanabe, S. Muramatsu, and R. Shibasaki, "Spatiotemporal route estimation consistent with human mobility using cellular network data," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 2013, pp. 267–272.

[25] L. Shi, G. Chi, X. Liu, and Y. Liu, "Human mobility patterns in different communities: a mobile phone data-based social network approach," *Annals of GIS*, vol. 21, no. 1, pp. 15–26, 2015.

[26] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1082–1090.

[27] M. Marjanović, L. Skorin-Kapov, K. Pripužić, A. Antonić, and I. P. Žarko, "Energy-aware and quality-driven sensor management for green mobile crowd sensing," *Journal of network and computer applications*, vol. 59, pp. 95–108, 2016.

[28] T. Hu, M. Xiao, C. Hu, G. Gao, and B. Wang, "A qos-sensitive task assignment algorithm for mobile crowdsensing," *Pervasive and Mobile Computing*, vol. 41, pp. 333–342, 2017.

[29] F. Yang, J.-L. Lu, Y. Zhu, J. Peng, W. Shu, and M.-Y. Wu, "Heterogeneous task allocation in participatory sensing," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[30] J. Wang, J. Tang, G. Xue, and D. Yang, "Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems," *Computer Networks*, vol. 115, pp. 100–109, 2017.

[31] 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 *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 1366–1374.

[32] K. Yi, R. Du, L. Liu, Q. Chen, and K. Gao, "Fast participant recruitment algorithm for large-scale vehicle-based mobile crowd sensing," *Pervasive and Mobile Computing*, vol. 38, pp. 188–199, 2017.

[33] D. Zhang, H. Xiong, L. Wang, and G. Chen, "Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014, pp. 703–714.

[34] H. Li, T. Li, and Y. Wang, "Dynamic participant recruitment of mobile crowd sensing for heterogeneous sensing tasks," in *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*. IEEE, 2015, pp. 136–144.

[35] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*. IEEE, 2015, pp. 55–62.

[36] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes, "icrowd: Near-optimal task allocation for piggyback crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010–2022, 2016.

[37] J. Wang, Y. Wang, D. Zhang, L. Wang, H. Xiong, A. Helal, Y. He, and F. Wang, "Fine-grained multitask allocation for participatory sensing with a shared budget," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1395–1405, 2016.

[38] J. Wang, Y. Wang, D. Zhang, F. Wang, Y. He, and L. Ma, "Psallocator: multi-task allocation for participatory sensing with sensing capability constraints," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, 2017, pp. 1139–1151.

[39] M. Xiao, J. Wu, H. Huang, L. Huang, and C. Hu, "Deadline-sensitive user recruitment for mobile crowdsensing with probabilistic collaboration," in *Network Protocols (ICNP), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–10.