Profile-Free and Real-Time Task Recommendation in Mobile Crowdsensing

Yang, G., Li, Y., He, X., Song, Y., Wang, J. & Liu, M.

DOI 10.1109/TCSS.2021.3073031
ESSN 2329-924X

Publisher: IEEE

Profile-Free and Real-Time Task Recommendation in Mobile Crowdsensing

Guisong Yang, *Member, IEEE*, Yanting Li, Xingyu He, Yan Song, *Member, IEEE*, Jiangtao Wang, *Member, IEEE*, and Ming Liu, *Senior Member, IEEE*

***Abstract*— As a key research issue in mobile crowdsensing (MCS), recent studies on task recommendation have begun to focus on recommending tasks to participants according to the learned participant preferences. The common drawbacks of these studies are that, on the one hand, the factors affecting participant preferences are predefined, which is not practical as the influential factors are quite complex and a full map of participant profiles needs to be preexisted. On the other hand, they do not consider how to update the recommendation dynamically. To overcome these drawbacks, a profile-free and real-time task recommendation method is proposed in this work. First, we apply the recommendation systems to MCS to realize profile-free task recommendations. Second, a participant-task-location tensor is constructed, based on which an improved tensor factorization method is presented to provide task recommendations for participants at a given location. Finally, we design a real-time update algorithm based on the idea of one update at a time to update task recommendation lists for participants in real time. Based on real-world trace data sets, extensive evaluations show that the proposed method has obvious advantages over other baselines in terms of accuracy and time cost.**

***Index Terms*— Mobile crowdsensing (MCS), profile-free, real-time, task recommendation.**

I. Introduction

With the popularity of intelligent mobile devices equipped with diverse sensors, mobile crowdsensing (MCS) has become an effective sensing paradigm [1]. Specifically, MCS refers to participants execute sensing tasks with the sensors in their intelligent devices, so as to collect information about the surrounding environment (such as air quality, traffic flow, and water pollution) [2]–[4]. MCS has been widely studied in recent years, with a large number of MCS platforms established, such as WAZE [5], Million-agents [6], Medusa [7], PRISM [8], and CrowdOS [9].

The connection between tasks and participants is crucial in MCS systems. According to different ways of connection, MCS can be divided into the push mode and the pull mode [10]. The push mode means that the platform designs task allocation strategies based on the optimization objective and allocates tasks to participants. In the pull mode, the platform first releases sensing tasks; then, participants select tasks according to their preferences. Although the pull mode is easy to implement, it encounters some problems in practical applications. For example, a large number of tasks, in reality, are apt to information overload. In this case, participants need to select tasks from a large number of tasks, leading to a poor participant experience and reducing the efficiency of task selection. To solve this problem, the task that participants are more likely to select should be placed in front of other tasks. The likelihood of participants selecting tasks is affected by participants' preferences and their location [11], [13]. Therefore, important work is to provide task recommendations to the participant based on the predicted likelihood of the participant selecting each task.

Most previous studies have used the logistic regression model [12]–[14] to learn participants' preferences to realize task recommendations. However, in these studies, the factors affecting participant preferences need to be predefined, which is not practical. On the one hand, the influential factors are quite complex; it is difficult to define all factors in advance. On the other hand, the full map of participant profiles corresponding to these factors needs to be preexisted. Thus, a profile-free approach is needed to realize task recommendations. In fact, recommendation systems can learn valuable patterns from complex historical data and help participants discover profitable items, without the need to define the factors affecting participant preferences in advance. In this article, we propose to apply the recommendation systems to MCS to realize location-based task recommendations. In order to improve the accuracy of task recommendation, the tensor factorization method is involved to learn the relationship among participants, tasks, and locations. However, the existing tensor factorization methods cannot be directly used in MCS

Manuscript received December 2, 2020; revised March 12, 2021; accepted April 8, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61802257 and Grant 61602305 and in part by the Natural Science Foundation of Shanghai under Grant 18ZR1426000 and Grant 19ZR1477600. *(Corresponding author: Yan Song.)*

Guisong Yang is with the Department of Computer Science and Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China, and also with the Shanghai Key Laboratory of Modern Optical System, Shanghai 200093, China (e-mail: gsyang@usst.edu.cn).

Yanting Li is with the Department of Computer Science and Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China (e-mail: 182590522@st.usst.edu.cn).

Xingyu He is with the College of Communication and Art Design, University of Shanghai for Science and Technology, Shanghai 200093, China (e-mail: xy_he@usst.edu.cn).

Yan Song is with the Department of Control Science and Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China (e-mail: sonya@usst.edu.cn).

Jiangtao Wang is with the Center for Intelligent Healthcare, Coventry University, Coventry CV1 5RW, U.K. (e-mail: jiangtao.wang@coventry.ac.uk).

Ming Liu is with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong 999077 (e-mail: eelium@ust.hk).

scenarios. In most MCS applications, participants are not required to explicitly rate the tasks that they have executed, which leads to difficulty in obtaining explicit feedback data. In comparison, the implicit feedback data are easy to obtain and carry rich information. Therefore, the tensor factorization method needs to adapt to the implicit feedback data in MCS. In addition, since the data in the tensor in MCS are nonnegative, such as the number of times that participants have executed each task at each location, the nonnegativity of the tensor should be guaranteed in the process of tensor factorization. Therefore, it is necessary to improve the existing tensor factorization method to adapt to the characteristics of MCS.

In addition, timeliness is also an important evaluation indicator in task recommendations. In MCS systems, the participants' historical data of task execution change with time. In order to improve the accuracy of task recommendations, the learned participants' preferences need to be updated timely with the real-time updating of historical feedback data so that the task recommendation list for participants can be updated accordingly. However, the existing studies have not considered this problem, which reduces the accuracy of task recommendations to a certain extent. In order to update the recommendation dynamically, one straightforward way is to use batch processing; that is, once the participant's historical data change, the platform will learn participants' preferences again based on all historical feedback data. However, the number of participants and tasks in MCS systems may be very large; that is, the scale of historical feedback data is very large, and it will be very expensive in terms of time and calculation. Therefore, a more efficient method to achieve real-time updates of the task recommendation list is of vital importance.

With the aforementioned research objective and challenges, a profile-free and real-time task recommendation method (PFRT-TR) is proposed, and the main contributions of this ARTICLE are summarized as follows.

1) In view of the drawbacks of previous studies that need to predefine the factors affecting participant preferences, we apply the recommendation systems to MCS to realize profile-free task recommendations.
2) We construct a participant-task-location tensor to describe their relationships. On this basis, considering the implicit feedback and nonnegativity characteristics of the data in the MCS scenarios, an improved tensor factorization method is presented to provide accurate task recommendations for the participants at a given location.
3) In order to update the participants' task recommendations dynamically, a lightweight real-time update algorithm is designed based on the idea of one update at a time, which can reduce the time cost and calculation consumption efficiently when updating the task recommendation list. In addition, the social relationship among participants is introduced in the algorithm to improve the accuracy of task recommendations.

The rest of this article is organized as follows. In Section II, the related works of task recommendation and recommendation systems are provided. In Section III, matrix factorization (MF) and tensor factorization are introduced. The real-time updating algorithm is designed in Section IV. Section V gives the performance evaluation. Finally, Section VI provides the conclusion.

II. Related Work

A. Task Recommendation in Mobile Crowdsensing

Task recommendation where the platform recommends tasks to participants in a certain order for them to choose from is a core issue in MCS. In recent years, studies increasingly focus on improving the accuracy of task recommendations. An effective task recommendation framework with win–win incentives is proposed for crowdsourcing systems. Through bipartite matching between the task and participant, both the requester and the participant benefit from the task recommendation [15]. However, this framework is limited by the fact that it requires explicit ratings of tasks by participants in the historical data, and most MCS applications are unable to offer explicit feedback. In order to solve this problem, implicit feedback is used for task recommendations. Karaliopoulos *et al.* [12] used logistic regression models and machine learning techniques to build personalized behavior decision-making profiles for participants based on their responses to past tasks with different attribute values. To overcome the limitations of the content-based method and the collaborative method, a hybrid recommendation approach is proposed in [13] to predict the probability of participants choosing each task. Wang *et al.* [14] exploited the content information to accurately model participants' preference on tasks and utilized the Logit model to integrate the heterogeneous factors into a single framework to predict the matching probability of each task–participant pair. Although these studies can achieve task recommendations, they need to predefine the factors affecting participant preferences, this is not practical as the influential factors are quite complex, and a full map of participant profiles needs to be preexisted. Therefore, we apply the recommendation systems to MCS to realize profile-free task recommendations.

In task recommendation, in addition to the participant's preference, participants' location also affects the selection of participants [11]. Therefore, participants' location should be considered when recommending tasks for participants to improve the accuracy of recommendation. In recent years, mobile location technology has been widely developed, which makes it easier to obtain location information to provide related services, thus effectively narrowing the gap between the real world and MCS services. Therefore, a series of location-related studies have been proposed.

Research on location can be roughly divided into the following aspects. First, considering the impact of location on task execution, some studies have been proposed to determine the final task allocation strategy based on location design optimization goals. A location-aware task allocation and routing strategy is proposed to minimize the total moving distances of all participants subject to several constraints (e.g., sensor and deadline) [16]. Tao and Song [17] studied the task allocation problem taking geographic distributions of tasks

into account, especially the clustering tasks. Gong *et al.* [18] studied the task allocation and path planning problem in mobile crowdsensing, which aims to maximize total task quality under constraints of participant travel distance budgets. Second, considering the privacy needs of participants in practical applications, some studies focus on location privacy protection strategies. Gao *et al.* [19] proposed a novel and efficient location privacy-preserving truth discovery (LoPPTD) mechanism, which can protect both location privacy and data privacy of participants. Li *et al.* [20] developed a privacy-preserving participant recruiting scheme for mobile crowdsensing, which guarantees the crowdsensing coverage while preserving participants' location differential privacy against a semihonest crowdsensing aggregator. Jin *et al.* [21] constructed a participant-centric location privacy trading framework ULPT to facilitate location privacy trading between participants and the platform. These studies consider the role of location factors in the MCS systems but ignore the influence of participants' preferences on decision-making, thus reducing the efficiency of task allocation to a certain extent. Unlike these studies, we recommend tasks to participants considering both location and preference.

In addition, previous studies do not consider how to update the task recommendation list dynamically, which makes the task recommendation unable to adapt to changes in historical data in time. To solve the abovementioned problems, a real-time update algorithm is designed to update participants' task recommendations in real time.

B. Recommendation Systems

Originating as a solution to information overload, recommendation systems are composed of participants, items, and corresponding ratings. Such systems can accurately learn participant preferences and improve personalized service, which has broad application prospects. So far, many models have been designed to improve the accuracy of recommendation, such as trace-norm regularization-based models [22], multidimensional probabilistic models [23], and latent factor (LF) models [24]. Among these models, LF models have been proven to have excellent performance and be able to reduce the cost of calculation. Therefore, LF models have received widespread attention, and the MF is the most commonly used model in LF. Due to its high precision and scalability, MF has been used to handle various data analysis tasks [25], [26].

Matrices with dimensions greater than or equal to three are called tensors, and the decomposition of tensors has important applications in signal processing [27], [28], data mining [29], and graph analysis [30]. Existing tensor factorization methods include HOSVD [31], the pairwise interaction approach [32], and [33]. However, these studies are not applicable to the implicit feedback and nonnegativity characteristics of the data in the MCS scenarios. In this article, an improved tensor factorization method is presented to adapt to the historical data with implicit feedback, which is more in line with the actual applications of MCS.

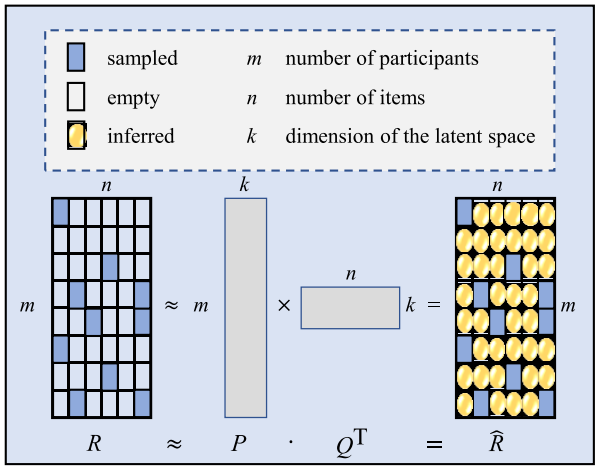

Fig. 1. MF diagram.

III. Preliminaries—Matrix Factorization and Tensor Factorization

In this section, we summarize the methods of MF and tensor factorization. For easily understanding, tensors are denoted by calligraphic uppercase letters (e.g., $\mathcal{A}$ and $\mathcal{B}$), matrices are denoted by uppercase letters (e.g., A and B), and scalars are denoted by lowercase letters (e.g., a and b).

A. Matrix Factorization

For a 2-D matrix, recommendation systems usually use MF to learn participant preferences. In MF, the objective matrix is decomposed into two low-rank matrices by mining potential information; then, the product of the two matrices is used to approximate the objective matrix.

For example, as shown in Fig. 1, the matrix R is a rating matrix, in which the elements represent the historical rating data of the participants on the items. In practical applications, participants usually only rate a few items, so the rating matrix R is a sparse matrix, that is, most of the element values in R are missing. Then, we use MF to decompose R into a low-rank participant LF matrix P and a low-rank item LF matrix Q and take the product of P and Q as the approximate matrix of R, so as to obtain a prediction rating matrix $\hat{R}$ without missing values.

MF is achieved through iteration with the goal of minimizing the cost function. The cost function is usually defined as the error between the predicted rating and the actual rating, which is expressed as

$$\arg\min_{P,Q} J(P,Q) = \frac{1}{2}\sum_{r_{i,j}\in\Lambda}\left(r_{i,j} - \sum_{s=1}^{k} p_{i,s}q_{j,s}\right)^2 + \frac{\lambda}{2}\sum_{s=1}^{k}\left(\sum_{i=1}^{m} {p_{i,s}}^2 + \sum_{j=1}^{n} {q_{j,s}}^2\right) \quad (1)$$

where $r_{i,j}$ denotes the actual rating of the participant p_i on item t_j, Λ represents the known entries of matrix R, P represents the participant LF matrix, and Q represents the item LF matrix. $p_{i,s}$ and $q_{j,s}$ represent the element in the LF matrices P and Q, respectively. The second half of

the formula is called the regularization term, and its purpose is to prevent overfitting.

B. Tensor Factorization

When the dimension N of a matrix is greater than 2, the matrix is called N-order tensor, which can be denoted as $R^{I_1 \times I_2 \times \cdots \times I_N}$. Accordingly, the elements in the tensor are expressed as $r_{i_1,i_2,\ldots,i_N}$.

For tensor factorization of the third-order tensor, [32] proposed a method to decompose the third-order tensor into three LF matrices and redefined the cost function as

$$\begin{aligned}&\arg\min_{P,Q,W} J(P,Q,W)\\&=\frac{1}{2}\sum_{i,j,c}\left(r_{i,j,c}-\sum_{s=1}^{k}\left(p_{i,s}q_{j,s}+q_{j,s}w_{c,s}+p_{i,s}w_{c,s}\right)\right)^2\\&\quad+\frac{\lambda}{2}\sum_{s=1}^{k}\left(\sum_{i=1}^{m}{p_{i,s}}^2+\sum_{j=1}^{n}{q_{j,s}}^2+\sum_{c=1}^{d}{w_{c,s}}^2\right)\end{aligned}\tag{2}$$

where $r_{i,j,c}$ represents the element in the third-order tensor, and $p_{i,s}$, $q_{j,s}$, and $w_{c,s}$ represent the element value in the LF matrix respectively. Similar to (1), the second half of the formula is the regularization term, which is used to prevent overfitting.

IV. Proposed Method

In order to provide real-time task recommendations to participants at a given location, in this work, a PFRT-TR is proposed. The method can be divided into the following four steps, as shown in Fig. 2. First, the platform counts the number of times that participants have executed each task at each location and establishes a third-order participant-task-location rating tensor $\mathcal{R}$. Second, the platform uses the tensor factorization method to decompose $\mathcal{R}$ to generate the initial LF matrices P, Q, and W. Third, the platform first calculates the predicted participant-task-location rating tensor $\hat{\mathcal{R}}$ by operating LF matrices P, Q, and W. The elements in $\hat{\mathcal{R}}$ represent the predicted likelihood that the participant will execute the task at the given location. Then, by sorting tasks according to the likelihood of the participant executing them at a certain location, from high to low, the platform puts ordered tasks into a task recommendation list. Fourth, each time the platform updates the rating tensor $\mathcal{R}$, it updates the task recommendation list correspondingly. To indicate these steps clearly, the detailed symbols and descriptions utilized in this work are presented in Table I.

Given a set of m participants $P = \{p_1, p_2, \ldots, p_m\}$, a set of n tasks $T = \{t_1, t_2, \ldots, t_n\}$, and a set of d locations $L = \{l_1, l_2, \ldots, l_d\}$ in the MCS systems, the third-order participant-task-location rating tensor is denoted as $\mathcal{R} = [r_{i,j,c}]_{m\times n\times d}$. In the tensor, $r_{i,j,c}$ is characterized by the number of times that participant p_i has executed task t_j at location l_c. $P_{m\times k}$, $Q_{n\times k}$, and $W_{d\times k}$ represent the participant LF matrix, the task LF matrix, and the location LF matrix, respectively.

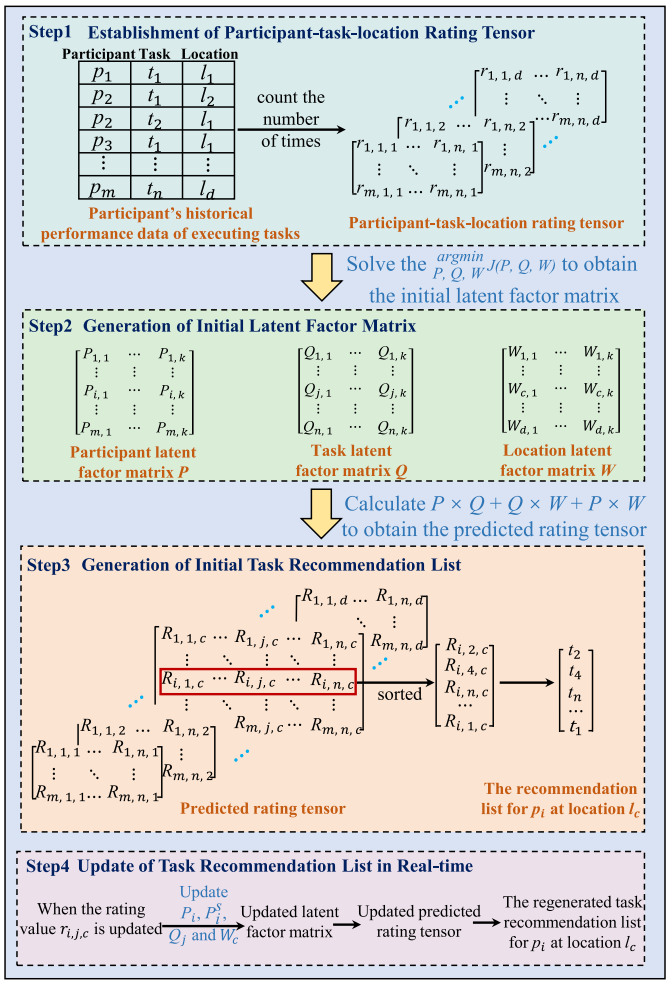

Fig. 2. Model of PFRT-TR.

TABLE I
Symbols and Descriptions

Symbol	Description
m	the number of participants
n	the number of tasks
d	the number of locations
k	the dimension of the latent space
α	the attenuation parameter
$\mathcal{R}_{m\times n\times d}$	the participant-task-location rating tensor
$P_{m\times k}$	the participant latent factor matrix
$Q_{n\times k}$	the task latent factor matrix
$W_{d\times k}$	the location latent factor matrix

A. Step 1: Establishment of Participant-Task-Location Rating Tensor

In MCS, it is necessary to collect feedback data of participants on tasks to learn participants' preferences. However, in most MCS applications, participants are not required to explicitly rate the tasks that they have executed, which leads to difficulty in obtaining explicit feedback data. In this case, implicit feedback data have advantages to be used for task recommendation: for one thing, the implicit feedback data of participants, which is composed of the number of times that they have executed each task at each location, can reflect their preferences to a certain extent; for another, the implicit

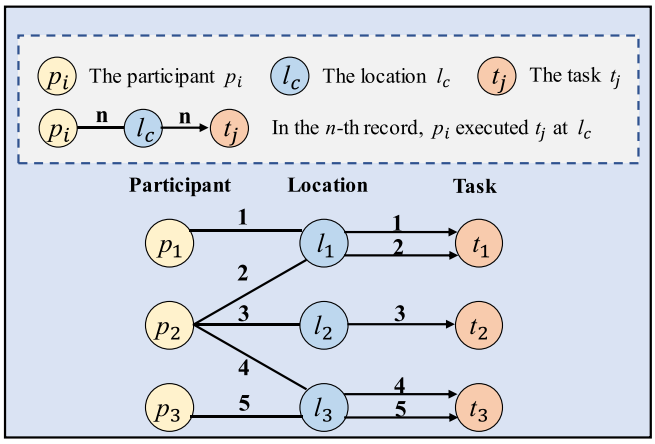

Fig. 3. Instance of participants executing tasks at the given location.

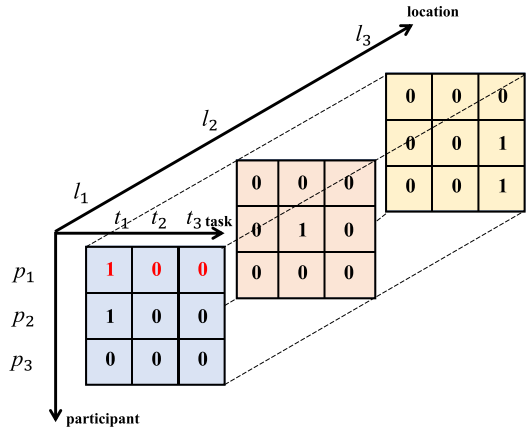

Fig. 4. Initial third-order participant-task-location tensor.

feedback data can be directly collected by the platform, so it is easy to obtain. In this article, these implicit feedback data are used to establish third-order participant-task-location rating tensor.

The participant p_i's historical records of executing task t_j at location l_c can be represented by a triple $\{p_i, t_j, l_c\}$. Then, the number of times $r_{i,j,c}$ that participant p_i has executed task t_j at location l_c can be obtained by accumulating the number of occurrences of these triples. Based on that, a third-order participant-task-location tensor $\mathcal{R}$ can be established. If privacy is a concern, the MCS platform only needs to know the encoded IDs of participants, not their physical identity [34]. Privacy is not our focus.

To facilitate the understanding of the process, an instance of establishing a participant-task-location rating tensor is shown in Figs. 3 and 4. We assume that there are three participants, three tasks, and three locations in an MCS system. In Fig. 3, the arrow lines with different numbers represent different records. For example, the line labeled 5 represents a record of participant p_3 executing task t_3 at location l_3. By accumulating the records in Fig. 3, we establish a third-order participant-task-location tensor $\mathcal{R}$ in Fig. 4. The three dimensions in $\mathcal{R}$ represent participants, tasks, and locations, respectively, and the elements in $\mathcal{R}$ represent the number of occurrences of records. For example, the row vectors marked red in Fig. 4 indicate that participant p_1 has executed tasks t_1, t_2, and t_3 at location l_1 for 1, 0, and 0 times, respectively.

It should be pointed out that the instance presented in this section is only used to explain the process of our method, so the data volume used is very small, and the data volume in the actual MCS applications is much larger than this.

B. Step 2: Generation of Initial Latent Factor Matrix

In this step, we decompose $\mathcal{R}$ to generate the initial LF matrix; one simplest way is to use the cost function in (2). However, this will reduce the accuracy because $\mathcal{R}$ is a tensor formed by implicit feedback data rather than the actual rating of the participants on the tasks and cannot be used to express participants' preferences on tasks directly. For example, if there is no record of a participant executing a task at a certain location in the historical data, it is hard to determine whether the participant is not interested in the task at that location or participant just does not see the task because of its low ranking in the task list. Therefore, we introduce the confidence degree $1+\alpha r_{i,j,c}$ into (3) to represent the reliability of using $r_{i,j,c}$ to represent the participant p_i's preference on task t_j at the location l_c, where α represents the attenuation parameter.

Furthermore, since the elements in $\mathcal{R}$ represent the number of times that the participant p_i has executed the task t_j at the location l_c, it is nonnegative in practical MCS applications. Thus, the cost function can be formulated by

$$\begin{aligned}&\arg\min_{P,Q,W} J(P,Q,W)\\&=\frac{1}{2}\sum_{i,j,c}1+\alpha_{r_{i,j,c}}\\&\quad\times\left(r_{i,j,c}-\sum_{s=1}^{k}p_{i,s}q_{j,s}+q_{j,s}w_{c,s}+p_{i,s}w_{c,s}\right)^2\\&\quad+\frac{\lambda}{2}\sum_{s=1}^{k}\left(\sum_{i=1}^{m}{p_{i,s}}^2+\sum_{j=1}^{n}{q_{j,s}}^2+\sum_{c=1}^{d}{w_{c,s}}^2\right)\\&\text{s.t. }\{p_{i,s},q_{j,s},w_{c,s}\}\geq 0\end{aligned}\tag{3}$$

where $J(P,Q,W)$ represents the cost function about the LF matrix P, Q, and W; $p_{i,s}$ represents the element in P; $q_{j,s}$ and $w_{c,s}$ represent the elements in Q and W respectively; and the constraint condition is to ensure that the variables $p_{i,s}$, $q_{j,s}$, and $w_{c,s}$ are nonnegative.

Next, we focus on solving (3) to obtain the initial LF matrix. By adopting the additive gradient descent (AGD) algorithm, we obtain the update rules shown in (4)–(6), at the bottom of the next page.

Note that there are negative terms in the above update rules, such as $-\zeta_{i,h}\gamma_{i,j,c,s,h}$, $-\zeta_{j,h}\xi_{i,j,c,s,h}$, and $-\zeta_{c,h}\pi_{i,j,c,s,h}$, where

$$\begin{aligned}&\gamma_{i,j,c,s,h}\\&=\left(\sum_{j,c}1+\alpha_{r_{i,j,c}}\sum_{s=1}^{k}p_{i,s}q_{j,s}+q_{j,s}w_{c,s}+p_{i,s}w_{c,s}\right)\\&\xi_{i,j,c,s,h}\qquad q_{j,h}+w_{c,h}\quad+\lambda p_{i,h}\end{aligned}\tag{7}$$

$$= \begin{pmatrix} \left(\sum_{i,c} 1+\alpha_{r_{i,j,c}} \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\right) \\ p_{i,h} + w_{c,h} + \lambda q_{j,h} \end{pmatrix} \quad (8)$$

$$\pi_{i,j,c,s,h} = \begin{pmatrix} \left(\sum_{i,j} 1+\alpha_{r_{i,j,c}} \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\right) \\ p_{i,h} + q_{j,h} + \lambda w_{c,h} \end{pmatrix}. \quad (9)$$

For the adopted AGD algorithm, in order to remove the negative terms in (4)–(6), we denote $\zeta_{i,h} = p_{i,h}/\gamma_{i,j,c,s,h}$, $\zeta_{j,h} = q_{j,h}/\xi_{i,j,c,s,h}$, and $\zeta_{c,h} = w_{c,h}/\pi_{i,j,c,s,h}$.

Finally, the update rules can be easily obtained as (10)–(12), shown at the bottom of the page.

It can be proved that (10)–(12) finally converge. Note that, according to the multiplication update rule, if P, Q, and W are initially nonnegative, they are always nonnegative.

According to the above rules, we can decompose the third-order participant-task-location tensor in the instance of step 1 and generate the initial LF matrix shown in Fig. 5.

C. Step 3: Generation of Initial Task Recommendation List

After generating the initial LF matrices P, Q, and W, a predicted participant-task-location rating tensor $\hat{\Re}$ can be established by calculating $\hat{\Re} = P \times Q + Q \times W + P \times W$. The element values in $\hat{\Re}$ are calculated by $\widehat{r_{i,j,c}} = \sum_{s=1}^{k}(p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s})$, which represents the predicted likelihood that the participant p_i executes the task t_j at the location l_c, where $p_{i,s}, q_{j,s}$, and $w_{c,s}$ are the element values of the LF matrix separately. On this basis, the tasks are sorted from high to low according to the likelihood of the participant executing them at a certain location, and a task recommendation list with ordered tasks can be generated for the participant.

$$\arg\min_{P,Q,W} J(P,Q,W) \overset{\text{AGD}}{\Rightarrow} i,j,c;h, \quad s \in \{1,2,\ldots,k\}$$

$$p_{i,h} \leftarrow p_{i,h} - \zeta_{i,h}\frac{\partial J(P,Q,W)}{\partial p_{i,h}}$$

$$\leftarrow p_{i,h} + \zeta_{i,h}\left(\begin{pmatrix}\sum_{j,c} 1+\alpha_{r_{i,j,c}} \left(r_{i,j,c} - \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\right)\end{pmatrix}\right) \quad (4)$$

$$\begin{pmatrix} q_{j,h} + w_{c,h} - \lambda p_{i,h} \end{pmatrix}$$

$$q_{j,h} \leftarrow q_{j,h} - \zeta_{j,h}\frac{\partial J(P,Q,W)}{\partial q_{j,h}}$$

$$\leftarrow q_{j,h} + \zeta_{j,h}\left(\begin{pmatrix}\sum_{i,c} 1+\alpha_{r_{i,j,c}} \left(r_{i,j,c} - \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\right)\end{pmatrix}\right) \quad (5)$$

$$\begin{pmatrix} p_{i,h} + w_{c,h} - \lambda q_{j,h} \end{pmatrix}$$

$$w_{c,h} \leftarrow w_{c,h} - \zeta_{c,h}\frac{\partial J(P,Q,W)}{\partial w_{c,h}}$$

$$\leftarrow w_{c,h} + \zeta_{c,h}\left(\begin{pmatrix}\sum_{i,j} 1+\alpha_{r_{i,j,c}} \left(r_{i,j,c} - \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\right)\end{pmatrix}\right) \quad (6)$$

$$\begin{pmatrix} p_{i,h} + q_{j,h} - \lambda w_{c,h} \end{pmatrix}$$

$$\arg\min_{P,Q,W} J(P,Q,W) \overset{\text{AGD}}{\Rightarrow} i,j,c;h, \quad s \in \{1,2,\ldots,k\}$$

$$p_{i,h} \leftarrow p_{i,h}\frac{\sum_{j,c} 1+\alpha_{r_{i,j,c}} r_{i,j,c} q_{j,h} + w_{c,h}}{\left(\begin{pmatrix}\sum_{j,c} 1+\alpha_{r_{i,j,c}} \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\end{pmatrix}\right)} \quad (10)$$

$$\begin{pmatrix} q_{j,h} + w_{c,h} + \lambda p_{i,h} \end{pmatrix}$$

$$q_{j,h} \leftarrow q_{j,h}\frac{\sum_{j,c} 1+\alpha_{r_{i,j,c}} r_{i,j,c} p_{i,h} + w_{c,h}}{\left(\begin{pmatrix}\sum_{i,c} 1+\alpha_{r_{i,j,c}} \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\end{pmatrix}\right)} \quad (11)$$

$$\begin{pmatrix} p_{i,h} + w_{c,h} + \lambda q_{j,h} \end{pmatrix}$$

$$w_{c,h} \leftarrow w_{c,h}\frac{\sum_{i,j} 1+\alpha_{r_{i,j,c}} r_{i,j,c} p_{i,h} + q_{j,h}}{\left(\begin{pmatrix}\sum_{i,j} 1+\alpha_{r_{i,j,c}} \sum_{s=1}^{k} p_{i,s}q_{j,s} + q_{j,s}w_{c,s} + p_{i,s}w_{c,s}\end{pmatrix}\right)} \quad (12)$$

$$\begin{pmatrix} p_{i,h} + q_{j,h} + \lambda w_{c,h} \end{pmatrix}$$

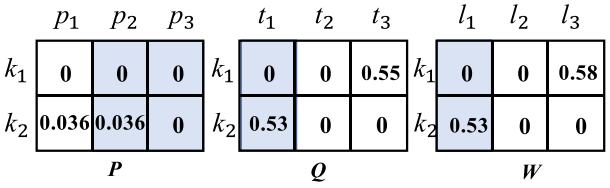

Fig. 5. Generated initial LF matrix.

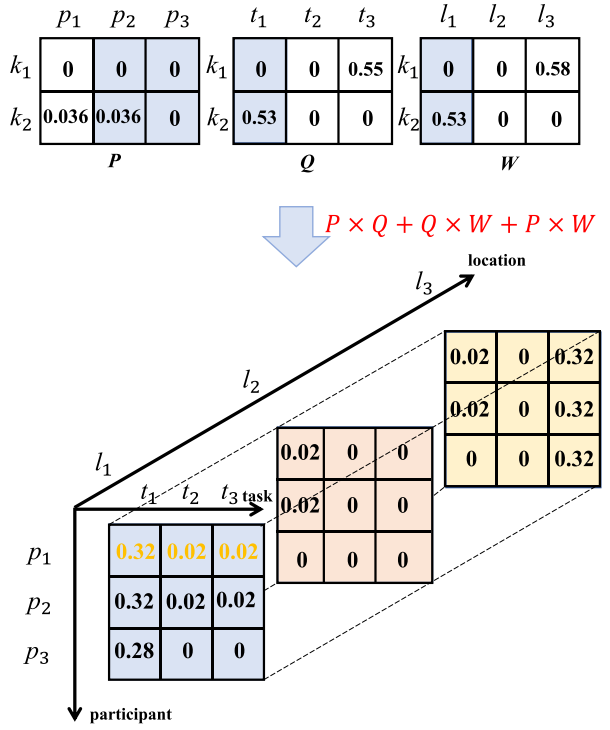

Fig. 6. Predicted third-order participant-task-location tensor based on LF matrices.

Fig. 6 shows the process of generating the predicted rating tensor based on the initial LF matrix obtained in Fig. 5. It can be seen that the likelihood of the participant p_1 executing tasks t_1, t_2, and t_3 at location l_1 is 0.32, 0.02, and 0.02, respectively. Sorting the three values of likelihood from high to low, we can generate the initial task recommendation list of the participant p_1 at location l_1 as $[t_1, t_2, t_3]$.

D. Step 4: Update of Task Recommendation List in Real Time

In order to predict the likelihood of the participant selecting tasks accurately so as to recommend suitable tasks to the participant, the platform will update the corresponding element values in $\mathcal{R}$ in real time each time the participant executing a task at a certain location; then, both the LF matrix and the task recommendation list for the participant should also be updated correspondingly.

After a participant executing a task, the platform will update the element representing the number of times that the participant has executed this task at this location, and the other elements in the tensor $\mathcal{R}$ do not need to be updated. In addition, the change of this element mainly affects a vector in the LF matrix. Therefore, different from the previous studies that update the entire LF matrix, we put forward the idea of one update at a time; that is, when the value

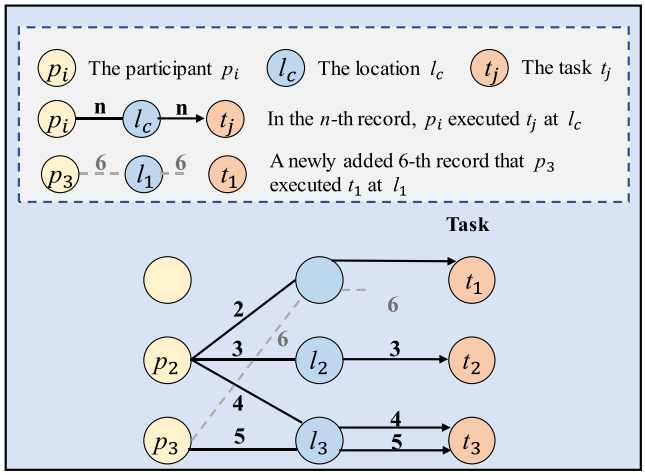

Fig. 7. Instance after adding a record that participant p_3 executes task t_1 at location l_1.

of the rating tensor $\mathcal{R}$ changes, only the corresponding LF vector influenced is updated. In addition, considering that social relations among participants will affect participants' willingness to select tasks, after a participant executes a task, the willingness of participants close to the participant to execute the same task may increase. Therefore, we propose to update the LF vector of other participants who are closely related to this participant at the same time. Here, we assume that a group of participants who often execute the same task in the same location have common interests, even if these participants do not have any physical or online connection. For a participant, other participants in the group are called his or her similar participants, and the similarity is represented by cosine similarity of the following formula:

$$\text{similarity}_{i,j} = \frac{\sum_{c=1}^{d} \frac{\text{TaskVector}_{i,c} \cdot \text{TaskVector}_{j,c}}{|\text{TaskVector}_{i,c}| \times |\text{TaskVector}_{j,c}|}}{d} \tag{13}$$

where $\text{TaskVector}_{i,c}$ is a vector whose length is equal to the number of tasks in the MCS systems, and the elements in the vector represent the number of times the participant p_i executed each task at the location l_c.

To sum up, when the values of $r_{i,j,c}$ change, we keep other vectors in the LF matrix unchanged and update P_i, P_i^s, Q_j, and W_c, so as to obtain the updated LF matrix, in which P_i^s represents the vector of similar participants of participant p_i. Based on the updated LF matrix, we can get the updated predicted rating tensor and can further regenerate the new task recommendation list according to step 3. Since the data volume of the LF vector is much smaller than that of the LF matrix, the time cost and resource consumption for updating the LF matrix are greatly reduced, and the requirement for real-time updating of the task recommendation list can be satisfied.

An instance of updating the task recommendation list is shown in Figs. 7 and 8. For example, as shown by the dotted arrow labeled 6, a record that participant p_3 executed task t_1 at location l_1 is added in Fig. 7. In order to update the task recommendation list correspondingly, the LF vector corresponding to p_3, t_1, l_1, and p_3's similar participants should be updated. In this case, we assume that only p_3's similar participants is needed. By using (14), the similarity between p_3 and p_1,

Algorithm 1 Profile-Free and Real-Time Task Recommendation

1 Input: number of participants m, number of tasks n, number of locations d, participant-task-location tensor $\mathcal{R}_{m\times n\times d}$, and parameters λ, α, k, $\mathcal{T}$
2 Output: participants' task recommendation list at the given location
3 Initialize: participant latent factor matrix $P_{m\times k}$, task latent factor matrix $Q_{n\times k}$, and location latent factor matrix $W_{d\times k}$
4 Initialize $t = 1$
5 //generate the initial latent factor matrix
6 while not converge and $t \leq \mathcal{T}$ do
7 for $i = 1$ to m do
8 update row vector of participant latent factor matrix $P[i]$
9 end for
10 for $j = 1$ to n do
11 update row vector of task latent factor matrix $Q[j]$
12 end for
13 for $c = 1$ to d do
14 update row vector of location latent factor matrix $W[c]$
15 end for
16 end while
17 // generate the initial task recommendation list
18 $\hat{\mathcal{R}} = P \cdot Q + Q \cdot W + P \cdot W$
19 for $i = 1$ to m do
20 for $c = 1$ to d do
21 Sort $\hat{\mathcal{R}}_{i,j,c}$ as Participant p_i's task recommendation list at the location l_c
22 end for
23 end for
24 //update the task recommendation list
25 while rating value $r_{i,j,c}$ changes do
26 find similar participant set S_i for participant p_i
27 update the corresponding value P_i, P_i^s, Q_j, and W_c
28 repeat line 17-23
29 end while

p_2 can be calculated, which are1/3 and 2/3, respectively, and then, p_2 is selected. Therefore, we use (11)–(13) to update the LF vector P_3, P_2, Q_1, and W_1 as the vectors of the marked color in Fig. 8. Based on this, we can regenerate a task recommendation list $[t_1, t_2, t_3]$ for participant p_3 at location l_1.

In this section, a PFRT-TR method is proposed. The specific process of the method is shown in Algorithm 1. In this algorithm, first, the confidence degree coefficient $c_{i,j,c}$ is introduced into the cost function of tensor factorization so that the tensor factorization method is more suitable for the historical data of the implicit feedback in MCS. Second, the formula of AGD is constrained to ensure the nonnegativity of the result. Third, an idea of one update at a time is proposed, and a real-time update algorithm is designed; based on it, it reduces the time cost and computational consumption of updating the recommendation list greatly and realizes the real-time updating of the task recommendation list. Finally, as social relationships have an impact on participants' willingness to select tasks, during the process of one update at a time, the LF vector of participant's similar participants should also be updated to improve the accuracy of updating the recommendation list.

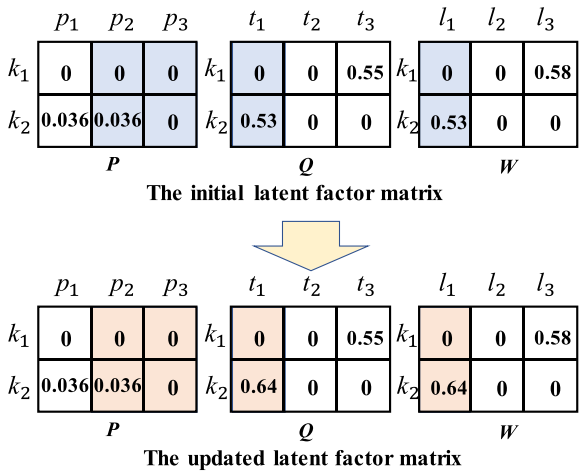

Fig. 8. Update the LF matrix.

E. Stability, Optimality, and Complexity Analysis

The stability, optimality, and complexity of the designed algorithm are analyzed as follows:

1) Stability Analysis: In line with [35] and [36], (10)–(12) can be proved to be ultimately convergent. Thus, the updated LF vectors obtained by (10)–(12) are unique; the tensor calculated by the LF vectors is also unique. In addition, as the task recommendation list is established based on this tensor, the final recommendation list obtained according to the algorithm is also unique. Therefore, the algorithm is stable.

2) Optimality Analysis: Compared with other studies, the designed algorithm is optimal in terms of accuracy and time cost, which is more in line with the actual needs of MCS applications. It is worth noting that, since we only update several vectors at a time, the final recommendation performance of the designed algorithm may decline over time compared to simply updating all vectors based on all historical data. Therefore, in order to further improve the accuracy of the recommendation result in practical applications, the system can update all vectors periodically after running the designed algorithm for a while.

3) Complexity Analysis: In Algorithm 1, the time complexity of initializing the three LF matrices in step 1 is $O[k(m + n + d)]$. The time complexity of generating the LF matrix in step 2 is $O(\mathcal{T}mndk)$. The time complexity of generating the initial task recommendation list in step 3 is $O(mndk + mk)$. The time complexity of updating the task recommendation list in real time is $O[k(nd + md + mn]$, which is lower than that of batch processing, that is, $O(mndk)$. As a result, the time complexity of Algorithm 1 is $O(\mathcal{T}mndk)$.

V. Performance Evaluation

A. Experimental Setting

1) Data Sets: The following two data sets are employed in our experiments to validate the proposed models.

TABLE II
DESCRIPTION OF DATA SETS

DATASETS	THE NUMBER OF PARTICIPANTS	THE NUMBER OF TASKS	THE NUMBER OF LOCATIONS
$\mathcal{D}1$	84	117	371
$\mathcal{D}2$	285	105	168

1) $\mathcal{D}1$ *(GeoLife GPS Trajectories) [37]:* The data set was collected by the Geolife project (Microsoft Research Asia) and recorded the movement trajectories of 182 participants, spanning from April 2007 to August 2012. The data set consists of 17 000+ pieces of GPS trajectory data, including longitude, latitude, and time, with more than 48 000 h in total. Since the data set is the trajectory data set, it does not include the historical data of specific task execution, so it is necessary to preprocess the data set. Specifically, we select part of the area in Beijing, which is in the northern latitude from 39.925 to 40.025 and eastern longitude from 116.29 to 116.37. We divide the area into 100 × 100 cells with equal size and generate 500 tasks randomly in cells every day. We suppose that, if the participant's trajectory is in the cell within that day, this participant can be deemed as having executed the task. Through processing, we finally extracted 216 245 data, including 84 participants, 117 tasks, and 371 locations.
2) $\mathcal{D}2$ *(FourSquare) [38]:* The data set consists of 1 021 970 check-in information at the location of participants. The check-in information includes data id, participant id, task id, longitude, latitude, and check-in time. In the data set, the number of participants is more than 2 153 000, and the number of locations is more than 1 143 000. All this information comes from the FourSquare application and is extracted through the public API. In this data set, we use the check-in record as a record of the participant's task execution. Through screening the data set, we extracted 622 841 data, including 285 participants, 105 tasks, and 168 locations.

For each data set, the first 20% of the data in the data set is selected as offline data, and the last 80% of the data is input as stream data. Table II presents the information about the data sets that we use.

2) Parameter Settings: In order to evaluate the performance of the proposed method, tenfold cross validation is used to set the value of the correlation coefficient for each method. First, an experiment to analyze the effect of k on accuracy is designed, as shown in Fig. 9, to assist both two data sets to set the appropriate value of the dimension k of the LF space. It can be seen that the accuracy increases with the increase in k, and the growth tends to be slow when k is greater than 11 and 10 in $\mathcal{D}1$ and $\mathcal{D}2$, respectively. In addition, time cost and resource consumption will increase with the increase in k. Therefore, in order to improve the accuracy and save the cost, k's in $\mathcal{D}1$ and $\mathcal{D}2$ are set to 11 and 10, respectively. Similarly, through the tenfold cross-validation method, the values of α in $\mathcal{D}1$ and $\mathcal{D}2$ are set to 0.2, and the values of λ_u and λ_i are set to 0.01.

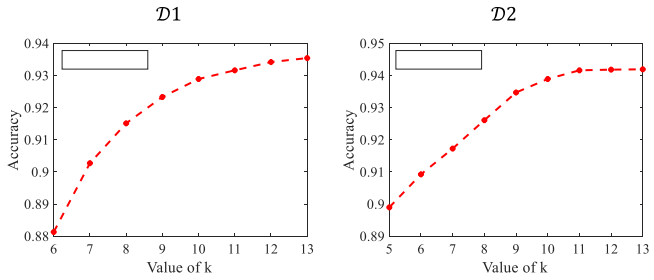

Fig. 9. Difference in accuracy due to different dimensions of the latent space on two data sets.

3) Baselines:: To emphasize the advantages of our method PFRT-TR, we compare them with eight baselines.

First, in order to verify the advantages of PFRT-TR in learning participant preferences, we compared PFRT-TR with the following baselines.

1) *FLTE [12]:* This method draws on logistic-regression techniques from machine learning to learn participants' individual preferences from historical data.
2) *UPR-E2 [13]:* This method defines each participant's preference for each task as a linear combination of the content-based characteristic and the collaborative-based characteristic.
3) *TPTO [14]:* This method treats the task–participant fitness prediction as a classification problem and utilizes the Logit model to learn the weight of each feature.

Different from PFRT-TR, they all need to predefine the factors affecting participant preferences.

Second, in order to verify the advantages of the improved tensor factorization method, we compared PFRT-TR with the following baselines.

1) *wLoc:* This method only considers the connection between participants and tasks when recommending tasks and does not consider the influence of location information.
2) *wPara:* This method does not consider the influence of implicit feedback and nonnegative characteristics of data in MCS scenarios.

Finally, in order to verify the effectiveness of the real-time update algorithm proposed in this article, we compare PFRT-TR with the following baselines.

1) *wReal:* This method is the retrained batch algorithm mentioned above, which updates the recommendation based on all historical feedback data.
2) *wSoc:* This method does not consider the influence of the participant's social relationship on the participant's willingness to execute the task.
3) *wUpd:* This method does not update the task recommendation list.

4) Evaluation Metrics:

1) *Accuracy*

$$\text{Accuracy} = \frac{\sum_{i=1}^{m}\sum_{c=1}^{d}\frac{\text{hitcount}_{i,c}}{\text{total recommend number}_{i,c}}}{m \times d} \quad (14)$$

where $\text{hitcount}_{i,c}$ represents the number of tasks in the recommendation list executed by the participant

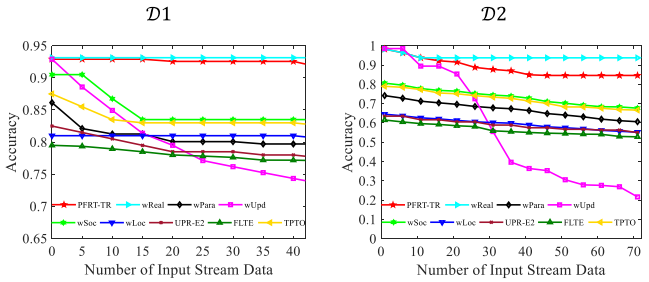

Fig. 10. Comparison of different algorithms in accuracy in two data sets.

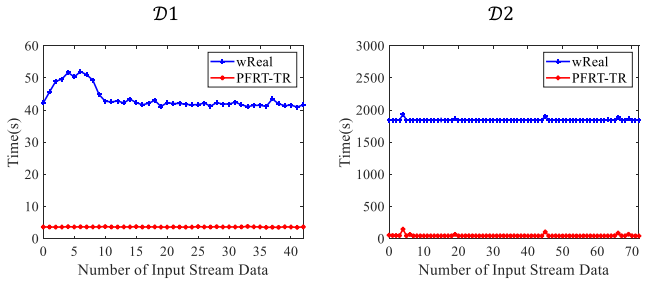

Fig. 11. Comparison of the time required for different update algorithms.

p_i at the location l_c, total recommend number$_{i,c}$ represents the number of tasks actually recommended to the participant p_i at the location l_c, m represents the number of participants, and d represents the number of locations in the MCS systems. Accuracy is characterized by the proportion of the number of tasks that participants have executed to the number of tasks recommended to participants, which is used to evaluate the performance of the task recommendation list.

2) *Ranking Metrics Normalized Discounted Cumulative Gain (NDCG)*

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} \tag{15}$$

where DCG is the discounted cumulative gain, and IDCG is the maximum DCG value under ideal conditions. Therefore, the value of NDCG is between (0, 1], which is used to measure whether the order of tasks in the task recommendation list is consistent with the participant's preferences.

B. Results and Discussion

To show the effectiveness and the superiority of the proposed method, we show their results on evaluation metrics compared with other baselines.

1) Recommendation Accuracy: Fig. 10 presents the accuracy results of the proposed PFRT-TR and other baselines. As can be seen from Fig. 10, PFRT-TR is superior to almost all other baselines in terms of accuracy.

First, PFRT-TR is more accurate than FLTE, UPR-E2, and TPTO, which proves that the recommendation systems have advantages in task recommendation. Then, PFRT-TR updates the task recommendation list in time with the change of participants' history data, so it can accurately adapt to the participants' preferences. Therefore, its accuracy is much higher than that of wUpd. In addition, PFRT-TR learns the relationships among participants, tasks, and locations, which is more in line with the actual applications of MCS. Therefore, it is superior to wLoc. In terms of parameter settings, wPara does not consider the impact of implicit feedback data, resulting in the inaccuracy of the learned participants' preferences. wSoc ignores the possible impact of the participant's social relationships on participants' willingness to select tasks, which reduces the performance of the recommendation to a certain extent. Finally, since wReal updates all values in the tensor during updating the task recommendation list, its accuracy is slightly higher than that of PFRT-TR. It is clear that the updated task recommendation list obtained by PFRT-TR has advantages in terms of accuracy in the MCS systems.

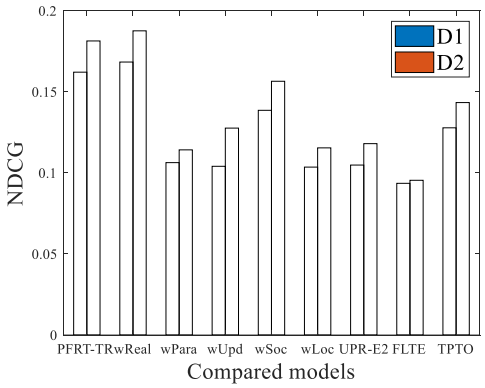

Fig. 12. Comparison of different methods in NDCG.

2) Time Cost: To illustrate the advantages of PFRT-TR in terms of time, we compare it with wReal, as shown in Fig. 11. Since PFRT-TR only needs to update a part of the LF vectors at one time, and the data volume to be updated by PFRT-TR is much smaller than that by wReal, the running time of PFRT-TR on the two data sets is much less than that of wReal, and the time spent on PFRT-TR is only about 0.08% that of wReal, which shows its efficiency in the real-time update.

3) NDCG: Before using NDCG to evaluate the ranking performance of the proposed method, the discounted cumulative gain DCG should be calculated first

$$\text{DCG} = \sum_{i=1}^{f} \frac{2^{\text{rel}_i} - 1}{\log_2 i + 1} \tag{16}$$

where f is the number of tasks recommended to participants, i.e., the length of the task recommendation list. In this article, we set $f = n$, where n is the number of tasks in the MCS systems. rel$_i$ is the flag indicating whether a participant executes the ith task in the participant's task recommendation list, with the value of 0 or 1. It can be seen from (16) that the higher the position of the task that the participant executes in the task recommendation list, the greater the DCG. Due to the value of IDCG is fixed for all participants, the greater the DCG, the greater the NDCG, and the better the ranking result.

Fig. 12 shows the PFRT-TR performs greater than other baselines in terms of NDCG except for wReal. However, the time cost of PFRT-TR only accounts for 0.08% of wReal's. Therefore, the ranking results of tasks in the task

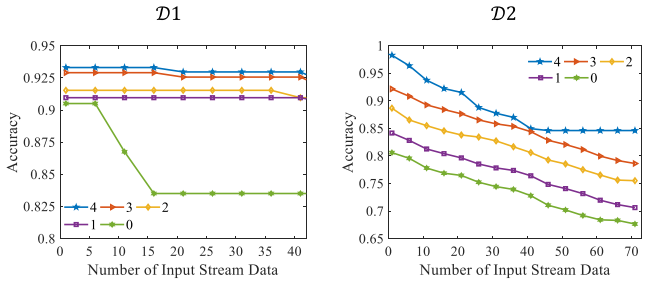

Fig. 13. Influence of social relationships on accuracy in two data sets.

recommendation list updated by PFRT-TR are more accurate and quickly meet the participant preferences.

4) Effect of Social Relationships: We conduct experiments on two data sets to verify the influence of social relationships on recommendation results, as shown in Fig. 13. Different lines in Fig. 13 indicate that the LF vector of different numbers of similar participants is updated. That is, the number in the legend indicates that, after a participant executes a task, not only the LF vector of the participant is updated but also the LF vector of this participant's corresponding number of similar participants is updated. As can be seen from Fig. 13, with the introduction of social relations, the accuracy rate is on the rise, so it is necessary to introduce social relations when updating the recommendation list.

VI. Conclusion

In this article, we proposed a PFRT-TR method that is divided into four steps. In the first step, a participant-task-location tensor is established to describe their relationships. In the second step, the tensor is decomposed using an improved tensor factorization method to generate the initial LF matrix. In the third step, a predicted tensor is calculated by operating the LF matrix, and on this basis, a task recommendation list is provided for the participants at a given location. Then, a profile-free task recommendation is realized through the above three steps. Finally, in the fourth step, each time the participant-task-location tensor is updated, the corresponding task recommendation list is updated to achieve real-time task recommendation. In future work, we will consider more factors that may affect task allocation in MCS systems and explore more sophisticated optimization methods and theoretical foundations.

References

[1] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 29–35, Aug. 2014.

[2] B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to mobile crowd sensing," in *Proc. IEEE Percom Workshops*, May 2014, pp. 593–598, doi: 10.1109/PerComW.2014.6815273.

[3] B. Guo *et al.*, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 1–31, Sep. 2015.

[4] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Active sparse mobile crowd sensing based on matrix completion," in *Proc. Sigmod*, Jun. 2019, pp. 195–210.

[5] *WAZE*. Accessed: Jun. 1, 2020. [Online]. Available: http://www.waze.com/

[6] *Millionagents*. Accessed: Jun. 1, 2020. [Online]. Available: http://www.millionagents.com/

[7] M. R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *Proc. ACM MobiSys*, Jun. 2012, pp. 337–350.

[8] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: Platform for remote sensing using smartphones," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2010, pp. 63–76.

[9] *CrowdOS*. Accessed: Jun. 1, 2020. [Online]. Available: http://www.crowdos.com/

[10] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: State-of-the-art and future opportunities," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3747–3757, Oct. 2018.

[11] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *Proc. IEEE 12th Int. Conf. Data Mining*, Dec. 2012, pp. 1038–1043.

[12] M. Karaliopoulos, I. Koutsopoulos, and M. Titsias, "First learn then earn: Optimizing mobile crowdsensing campaigns through data-driven user profiling," in *Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2016, pp. 271–280.

[13] F. Wu, S. Yang, Z. Zheng, S. Tang, and G. Chen, "Fine grained user profiling for personalized task matching in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, early access, May 11, 2020, doi: 10.1109/TMC.2020.2993963.

[14] Z. Wang *et al.*, "Towards personalized task-oriented worker recruitment in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 2080–2093, May 2021.

[15] W. Tang, K. Zhang, J. Ren, Y. Zhang, and X. (Sherman) Shen, "Privacy-preserving task recommendation with win-win incentives for mobile crowdsourcing," *Inf. Sci.*, vol. 527, pp. 477–492, Jul. 2020.

[16] S. Akter and S. Yoon, "Location-aware task assignment and routing in mobile crowd sensing," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2020, pp. 51–53.

[17] X. Tao and W. Song, "Location-dependent task allocation for mobile crowdsensing with clustering effect," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1029–1045, Feb. 2019.

[18] W. Gong, B. Zhang, and C. Li, "Location-based online task assignment and path planning for mobile crowdsensing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1772–1783, Feb. 2019.

[19] J. Gao, S. Fu, Y. Luo, and T. Xie, "Location privacy-preserving truth discovery in mobile crowd sensing," in *Proc. 29th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2020, pp. 1–9.

[20] L. Li, X. Zhang, R. Hou, H. Yue, H. Li, and M. Pan, "Participant recruitment for coverage-aware mobile crowdsensing with location differential privacy," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[21] W. Jin, M. Xiao, L. Guo, L. Yang, and M. Li, "ULPT: A user-centric location privacy trading framework for mobile crowd sensing," *IEEE Trans. Mobile Comput.*, early access, Feb. 9, 2021, doi: 10.1109/TMC.2021.3058181.

[22] N. Srebro and R. R. Salakhutdinov, "Collaborative filtering in a non-uniform world: Learning with the weighted trace norm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2056–2064.

[23] W. Chu and Z. Ghahramani, "Probabilistic models for incomplete multi-dimensional arrays," *Artif. Intell. Statist.*, vol. 5, pp. 89–96, Apr. 2009.

[24] Y. Chen, X. Li, and S. Zhang, "Structured latent factor analysis for large-scale data: Identifiability, estimability, and their implications," *J. Amer. Stat. Assoc.*, vol. 115, no. 532, pp. 1756–1770, Oct. 2020.

[25] X. Luo, M. Zhou, S. Li, L. Hu, and M. Shang, "Non-negativity constrained missing data estimation for high-dimensional and sparse matrices from industrial applications," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 1844–1855, May 2020.

[26] X. Luo, H. Wu, H. Yuan, and M. Zhou, "Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 1798–1809, May 2020.

[27] C. Qian, X. Fu, N. D. Sidiropoulos, and Y. Yang, "Tensor-based channel estimation for dual-polarized massive MIMO systems," *IEEE Trans. Signal Process.*, vol. 66, no. 24, pp. 6390–6403, Dec. 2018.

[28] D. S. Rocha, C. A. R. Fernandes, and G. Favier, "MIMO multi-relay systems with tensor space-time coding based on coupled nested tucker decomposition," *Digit. Signal Process.*, vol. 89, pp. 170–185, Jun. 2019.

[29] R. Sandhu, N. Kaur, S. K. Sood, and R. Buyya, "TDRM: Tensor-based data representation and mining for healthcare data in cloud computing environments," *J. Supercomput.*, vol. 74, no. 2, pp. 592–614, Feb. 2018.

[30] B. Jiang, C. Ding, J. Tang, and B. Luo, "Image representation and learning with graph-Lapxlacian tucker tensor decomposition," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1417–1426, Apr. 2019.

[31] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.
[32] C. Charu, "Context-sensitive recommender systems," in *Recommender Systems*. New York, NY, USA: Springer, 2016, pp. 255–281.
[33] B. Hidasi and D. Tikk, "Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, Jun. 2012, pp. 67–82.
[34] Z. Wang *et al.*, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 6, pp. 1330–1341, Jun. 2019.
[35] J. Zhao, S. Yang, H. Huo, Q. Sun, and X. Geng, "TBTF: An effective time-varying bias tensor factorization algorithm for recommender system," *Appl. Intell.*, Jan. 2021.
[36] X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, and K. Huang, "Block-randomized stochastic proximal gradient for low-rank tensor factorization," *IEEE Trans. Signal Process.*, vol. 68, pp. 2170–2185, Mar. 2020.
[37] Y. Zheng, L. Zhang, X. Xie, and W. Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proc. Int. Conf. World Wide Web (WWW)*, Madrid, Spain, Apr. 2009, pp. 791–800.
[38] M. Sarwat, "Sindbad: A location-based social networking system," in *Proc. ACM Sigmod Int. Conf. Manage. Data*, May 2012, pp. 649–652.

Guisong Yang (Member, IEEE) received the Ph.D. degree in control theory and control engineering from Tongji University, Shanghai, China, in 2013.

He was a Research Scholar with Michigan State University, East Lansing, MI, USA, from 2009 to 2011. He is currently an Associate Professor with the Department of Computer Science and Engineering, University of Shanghai for Science and Technology, Shanghai. His research interests include the Internet of Things and pervasive computing, delay-tolerant and opportunistic networks, and mobile crowdsensing.

Dr. Yang is a member of ACM.

Yanting Li received the B.S. degree in information management and information system from Jiangxi Agricultural University, Nanchang, China, in 2017. She is currently pursuing the master's degree in computer science and technology with the University of Shanghai for Science and Technology, Shanghai, China.

Her current research interests are mobile crowdsensing and recommendation systems.

Xingyu He received the Ph.D. degree in control theory and control engineering from Tongji University, Shanghai, China, in 2017.

She is currently an Assistant Professor with the College of Communication and Art Design, University of Shanghai for Science and Technology, Shanghai. Her research interests include wireless sensor networks and pervasive computing, delay-tolerant networks, incentive schemes, swarm intelligence, and mobile crowdsensing.

Yan Song (Member, IEEE) received the B.Sc. degree in materials science and engineering from Jilin University, Changchun, China, in 2001, the M.Sc. degree in applied mathematics from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in control theory and control engineering from Shanghai Jiao Tong University, Shanghai, China, in 2013.

From 2016 to 2017, she was a Visiting Scholar with the Department of Computer Science, Brunel University London, Uxbridge, U.K. She is currently an Associate Professor with the Department of Control Science and Engineering, University of Shanghai for Science and Technology, Shanghai. She has authored around 50 articles in refereed international journals. Her research interests include model predictive control, machine learning, and data analysis.

Dr. Song is a very active reviewer for many international journals.

Jiangtao Wang (Member, IEEE) received the Ph.D. degree from Peking University, Beijing, China, in 2015.

He was a Lecturer with the School of Computing and Communications, Lancaster University, Lancaster, U.K. He is currently a Tenured Associate Professor with the Center for Intelligent Healthcare, Coventry University, Coventry, U.K. His research interests include mobile and pervasive computing, crowdsensing/crowdsourcing, and the Internet of Things (IoT).

Ming Liu (Senior Member, IEEE) received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005, and the Ph.D. degree from the Department of Mechanical and Process Engineering, ETH Zürich, Zürich, Switzerland, in 2013, supervised by Prof. R. Siegwart.

He is currently with the ECE Department, the CSE Department, and the Robotics Institute, The Hong Kong University of Science and Technology, Hong Kong. His research interests include dynamic environment modeling, deep learning for robotics, 3-D mapping, machine learning, and visual control.

Dr. Liu received twice the Innovation Contest Chunhui Cup Winning Award, in 2012 and 2013 and the Wu Weijun AI Award, in 2016. He was the Program Chair of the IEEE-RCAR 2016, the Program Chair of the International Robotics Conference, Foshan, China, in 2017, and the Conference Chair of ICVS 2017.