Preference-Aware Task Assignment in Spatial Crowdsourcing

Yan Zhao,[1] **Jinfu Xia,**[1] **Guanfeng Liu,**[2] **Han Su,**[3] **Defu Lian,**[3] **Shuo Shang,**[4] **Kai Zheng**[3,5] *
[1]School of Computer Science and Technology, Soochow University, China
[2]Macquarie University, Australia
[3]University of Electronic Science and Technology of China, China
[4]King Abdullah University of Science and Technology, Saudi Arabia
[5]Youedata Research, Beijing, China
zhaoyan@suda.edu.cn, viing937@gmail.com, guanfeng.liu@mq.edu.au
hansu@uestc.edu.cn, {dove.ustc, jedi.shang}@gmail.com, zhengkai@uestc.edu.cn

Abstract

With the ubiquity of smart devices, Spatial Crowdsourcing (SC) has emerged as a new transformative platform that engages mobile users to perform spatio-temporal tasks by physically traveling to specified locations. Thus, various SC techniques have been studied for performance optimization, among which one of the major challenges is how to assign workers the tasks that they are really interested in and willing to perform. In this paper, we propose a novel preference-aware spatial task assignment system based on workers' temporal preferences, which consists of two components: *History-based Context-aware Tensor Decomposition (HCTD) for workers' temporal preferences modeling* and *preference-aware task assignment*. We model worker preferences with a three-dimension tensor (worker-task-time). Supplementing the missing entries of the tensor through HCTD with the assistant of historical data and other two context matrices, we recover worker preferences for different categories of tasks in different time slots. Several preference-aware task assignment algorithms are then devised, aiming to maximize the total number of task assignments at every time instance, in which we give higher priorities to the workers who are more interested in the tasks. We conduct extensive experiments using a real dataset, verifying the practicability of our proposed methods.

Introduction

Spatial Crowdsourcing (SC) is a recently proposed concept, which employs smart device carriers as workers to physically move to some specified locations and accomplish spatial tasks, such as taking photos, monitoring traffic condition and reporting local hot spot.

Most existing research focuses on the task assignment (Deng, Shahabi, and Zhu 2015; Cheng et al. 2015a; Alt et al. 2010; Tong et al. 2016a; 2016b; Song et al. 2017; Li, Yiu, and Xu 2015; Cheng et al. 2017; Zhao et al. 2017; Tong et al. 2018a; 2017; 2016b; 2018b), which aims to maximize the total number of completed tasks (Kazemi and Shahabi 2012), the number of performed tasks for a worker with an optimal schedule (Deng, Shahabi, and Demiryurek 2013), or the diversity score of assignments (Cheng et al. 2015b). An implicit assumption shared by these work is that the workers are willing to perform the tasks assigned to them. In practice, however, different task-performing intentions and preferences can lead to different types of behaviors. For example, two workers with different preferences for the same category of tasks may exhibit different behaviors: one is willing to report a hot spot for its popularity, while the other may not due to its complexity. Actually, a worker is unlikely to honestly and promptly complete the assigned tasks when she is not interested in them, which cannot guarantee the quality of task results. Therefore the key to control quality for task accomplishment is how to accurately capture worker preferences in her task-performing context. Among these existing contextual dimensions, time information, especially temporal dynamics, is of great importance since the characteristics of worker preferences with respect to the task types may change over the time of day. For instance, a worker is happy to report promotion activities of a shopping center during her lunch break but will definitely refuse to do it in her working hours. Therefore, incorporating temporal dynamics in worker preferences can improve the accuracy of spatial task assignment.

Several previous approaches infer worker preferences from past task-performing patterns or explicit feedbacks (Ambati, Vogel, and Carbonell 2011; Buchholz and Latorre 2011; Yuen, King, and Leung 2012). However, they fail to effectively incorporate temporal dynamics and workers' historical task-performing records. The overall task-performing behavior of a worker may be determined by her long-term interest. But at any given time, a worker is also affected by her instant preference due to transient events, such as the tasks' publishing and performing condition in the current time. In addition, the above methods are not able to make suitable task assignment since the task-performing data is extremely sparse and there exists cold start problem (no historical task-performing records for new workers or new tasks). Lastly, we are not aware of any existing task assignment techniques that consider the temporal dynamics in workers' preferences, which can be a key factor for improving the quality of task assignment in spatial crowdsourcing.

To address these challenges, we propose a Preference-

*Corresponding author: Kai Zheng.

aware Task Assignment (PTA) framework, based on sparse task-performing records generated in the recent time slots as well as in history. The framework is comprised of two primary components. First, we model different workers' preferences on different categories of tasks in different time slots with a three dimensional tensor. Supplementing the missing entries of tensor through History-based Context-aware Tensor Decomposition (HCTD) with the aid of workers' task-performing history and two context matrices, we recover worker preferences for different categories of tasks in different time slots. Secondly, we design three algorithms to maximize the overall task assignments by giving higher priorities to workers who are more interested in the tasks at every time instance.

In summary, our work has four primary contributions:

1) To the best of our knowledge, this is the first work in SC that considers workers' temporal preferences and performs task assignment based on these temporal preferences.

2) To address the task-performing data sparsity and cold start problem of individual workers, we propose a Temporal Preferences Modeling (TPM) method, which learns workers' temporal preferences from the task-performing data generated in the recent time slots as well as in history through HCTD with the assistant of other two context matrices.

3) We propose three alternative task assignment algorithms to solve our proposed problem. The first approach, Preference-aware Task Assignment (PTA) algorithm, follows the optimization strategy by maximizing the overall task assignments and giving higher priorities to workers with higher preferences for tasks at every time instance. The rest two algorithms, namely Spatial-weighted Preference-aware Task Assignment (SPTA) algorithm and Temporal-weighted Preference-aware Task Assignment (TPTA) algorithm, improve PTA by considering workers' travel cost and tasks' expiration time respectively.

4) Extensive experiments are conducted to verify the effectiveness of the proposed methods on a real dataset.

Problem Definition

Definition 1 (Spatial Task) *A spatial task, denoted by* $s =< s.l, s.p, s.\phi, s.c, s.maxW >$*, is a task to be performed at location* $s.l$*, published at time* $s.p$*, and will expire at* $s.p + s.\phi$*, where* $s.l : (x, y)$ *is a point in the 2D space. Each task* s *is also labelled with a category* $s.c$ *(e.g., taking photos, reporting local hot spot) and* $s.maxW$ *is the maximum number of workers allowed to be assigned to perform* s *at the same time instance.*

For simplicity, we assume that the processing time of each task is 0, which means that a worker will go to the next task upon arriving the location of the current task.

Definition 2 (Worker) *A worker,* $w =< w.l, w.r >$*, is a carrier of a mobile device who volunteers to perform spatial tasks. A worker can be in an either online or offline mode. A worker is online when she is ready to accept tasks. An online worker is associated with her current location* $w.l$ *and her reachable circular range with* $w.l$ *as the center and* $w.r$ *as the radius, where* w *can accept assignment of spatial tasks.*

In our model, a worker can handle only one task at a certain time instance, which is reasonable in practice. Once the server assigns a task to a worker, the worker is considered being offline until she completes the assigned task.

Definition 3 (Task-performing History) *Given a worker* w *who has performed* n *tasks in a time period, we define her task-performing history as a task set,* $S_w = \{(s_1, t^a_{s_1}, t^d_{s_1}), ...,(s_n, t^a_{s_n}, t^d_{s_n})\}$*, with each triplet* $(s_i, t^a_{s_i}, t^d_{s_i})$ *comprising the performed task* s_i*, worker's arrival time* $t^a_{s_i}$ *and departure time* $t^d_{s_i}$ *at the location of task* s_i*.*

For brevity, we simplify $S_w = \{(s_1, t^a_{s_1}, t^d_{s_1}), ..., (s_n, t^a_{s_n}, t^d_{s_n})\}$ as $S_w = \{s_1, ..., s_n\}$.

Definition 4 (Frequency-based Worker Preference) *Given a task category* c *and the task-performing history of worker* w*, we define the frequency-based preference of worker* w *in task category* c *in a certain time slot* $\mathbb{T}$*, denoted by* $P^{\mathbb{T}}_w(c)$*, as the ratio of tasks in category* c *to the total tasks that worker* w *has performed during* $\mathbb{T}$*, i.e.,*

$$P^{\mathbb{T}}_w(c) = \frac{\sum_{s_i \in S_w} \eta(s_i.c)}{N^{\mathbb{T}}(S_w)} \tag{1}$$

$$\eta(s_i.c) = \begin{cases} 1, s_i.c = c \text{ and } [t^a_{s_i}, t^d_{s_i}] \cap \mathbb{T} \neq \emptyset \\ 0, otherwise \end{cases}$$

where $N^{\mathbb{T}}(S_w)$ *is the number of tasks performed by* w *in a certain time slot* $\mathbb{T}$*.*

In the rest of the paper, we will use *worker preference* and *frequency-based worker preference* interchangeably when the context is clear.

Definition 5 (Spatial Task Assignment Instance Set) *Given the online worker set* $W_i = \{w_1, w_2, ...\}$ *and available task set* $S_i = \{s_1, s_2, ...\}$ *at time instance* t_i*, we define* A_i *as the spatial task assignment instance set at time* t_i*.* A_i *consists of a set of tuples of form* $< w, s >$*, where a spatial task* s *is assigned to worker* w*, satisfying all the workers' and tasks' constraints. We use* $|A_i|$ *to denote the number of task assignments at time instance* t_i*.*

Problem Statement: Given a set of online workers W_i and a set of available tasks S_i at the current time instance t_i on a SC platform, our problem is to find an allocation between the workers and tasks to maximize the total number of task assignments (i.e., $|A_i|$) by considering workers' temporal preferences of tasks at time instance t_i.

Framework Overview

Our framework (see Figure 1) is comprised of two major parts: 1) Temporal Preferences Modeling (TPM) for workers using History-based Context-aware Tensor Decomposition (HCTD); and 2) Preference-aware Task Assignment (PTA) based on workers' temporal preferences.

The TPM procedure constructs a 3D tensor $\mathcal{X}$ based on workers' recent and historical task-performing data, where the three dimensions stand for workers, task categories and time slots, respectively. Each entry is the preference of a particular worker for a particular task category in a certain

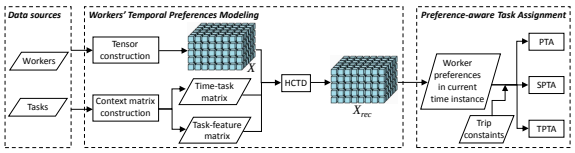


Figure 1: Framework of Our Model

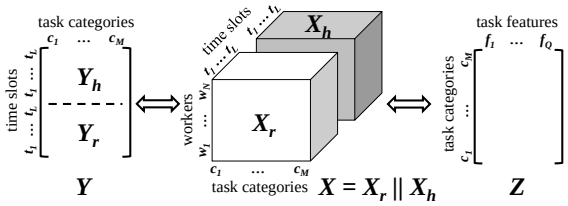


Figure 2: Workers' Temporal Preferences Modeling

time slot. Meanwhile, we build up two context matrices. One is the time-task matrix with two dimensions respectively standing for time slots and task categories, in which each entry is the number of times that the tasks in the corresponding category have been performed in a time slot. The other is the task-feature matrix whose values are extracted from historical worker profile. With the aid of the above two matrices, the missing entries of tensor $\mathcal{X}$ can be filled by HCTD. Then we can infer workers' preferences on all types of tasks in the current time. The idea behind it is that workers with similar contexts could have similar preferences. The context matrices reveal this inherent similarity and possess a much higher proportion of non-zero entries than $\mathcal{X}$, which can effectively reduce decomposition error and improve inference accuracy.

In the PTA phase, by considering trip constraints including workers' reachable region and tasks' expiration time, we optimize the task assignment based on workers' current preferences at every instance of time, and propose three algorithms, i.e., Preference-aware Task Assignment (PTA) algorithm, Spatial-weighted Preference-aware Task Assignment (SPTA) algorithm, and Temporal-weighted Preference-aware Task Assignment (TPTA) algorithm.

Workers' Temporal Preferences Modeling

In this section, we model workers' temporal preferences, which consists of three parts: 1) workers' temporal preferences (including recent and historical preferences) tensor construction; 2) context matrix construction that captures the temporal correlation of task-performing conditions as well as the similarity between different task categories; and 3) history-based context-aware tensor decomposition and completion, which decomposes the tensor with the aid of workers' historical preferences tensor and context matrices collaboratively, achieving a higher accuracy for workers' temporal preferences modeling.

Workers' Temporal Preferences Tensor Construction

In this section, we build a worker-task-time tensor, $\mathcal{X}_r \in \mathbb{R}^{N\times M\times L}$, based on the task-performing records in the most recent L time slots, to model the workers' temporal preferences for different categories of tasks, as illustrated in Figure 2. The tensor consists of three dimensions, i.e., workers, task categories and time slots, and each entry $\mathcal{X}_r(i,j,k)=e$ denotes the i-th worker's preference e on the j-th task category in time slot k (e.g., $10:00am-11:00am$). Obviously, there exists missing entries in tensor $\mathcal{X}_r$. Once the missing entries are inferred from other non-zero entries, we can obtain all the workers' preferences on any tasks in all the L time slots.

However, the tensor is over sparse with large quantities of missing entries since only a few tasks can be performed by an individual worker in a short time period. It is not accurate enough to decompose $\mathcal{X}_r$ solely based on its own non-zero entries, so we introduce another tensor, $\mathcal{X}_h$, based on the historical task-performing records over a longer period of time (e.g., one month) aggregated by the corresponding time slots from 1 to L, which has the same structure as $\mathcal{X}_r$ shown in Figure 2. Clearly, $\mathcal{X}_h$, representing the historical task-performing patterns and workers' long-term interests, is much denser than $\mathcal{X}_r$. The error of supplementing $\mathcal{X}_r$ can be greatly reduced by decomposing $\mathcal{X}_r$ and $\mathcal{X}_h$ together.

Context Matrix Construction

For more effective decomposition of the tensor $\mathcal{X}_r$, we also construct another two matrices, i.e., time-task matrix Y and task-feature matrix Z.

Time-task matrix. Matrix Y consists of Y_r and Y_h, capturing the temporal correlation in terms of the distribution of task-performing conditions over different task categories, in which each row denotes a time slot and each column denotes a task category. In our work, Y_r and Y_h respectively represent the recent and historical task-performing conditions in the same span of time of day. An entry of $Y_r \in \mathbb{R}^{L\times M}$, $Y_r(k,j)$, represents the number of times that the tasks of category j have been performed in time slot k. Consequently, the similarity of two different rows indicates the correlation of task-performing flows between two time slots. A worker may perform some similar tasks in time slot t_k and t_g since these two time slots share a similar worker task-performing pattern. For instance, the task-performing behaviours of a worker might be similar at 10:00am-11:00am and 2:00pm-3:00pm, since she is likely to stay at her workplace and willing to perform some simple tasks, which do not affect her normal duties. Moreover, Y_r can be more dense as its entries are aggregated from all the workers, therefore can help reduce the error of decomposing $\mathcal{X}_r$. Y_h has the same structure as Y_r, storing the number of times that the tasks with different categories have been performed in different time slots based on workers' long term task-performing history.

Task-feature matrix. Matrix $Z \in \mathbb{R}^{M\times Q}$ captures the similarity between different task categories by storing the task features of each category. Many features can be extracted based on different application scenarios, such as task popularity, task difficulty, task risk level, skill requirement,

and statistical information derived from historical worker profile. Each entry $Z(j, f)$ of Z represents the f-th feature of task category j. The value of $Z(j, f)$ can be a real value which indicates the weight of the feature for the task category.

Tensor Decomposition and Completion

Due to our goal of modeling workers' temporal preferences, we need to estimate the missing entries of $\mathcal{X}_r$. A straightforward solution is leveraging tucker decomposition model, which is generally applied to higher-order principal component analysis (Kolda and Bader 2009). It decomposes a tensor into a core tensor multiplied (or transformed) by a few matrices, based on the tensor's non-zero entries. However, decomposing $\mathcal{X}_r$ solely cannot get accurate enough results since it is over sparse. For instance, when using one-day task-performing records in our dataset and setting 1 hour as a time slot, only 0.15% entries of $\mathcal{X}_r$ are non-zero.

To achieve a high accuracy of preference estimation, we combine $\mathcal{X}_r$ and $\mathcal{X}_h$ (i.e., $\mathcal{X} = \mathcal{X}_r||\mathcal{X}_h$) together and then decompose $\mathcal{X} \in \mathbb{R}^{N\times M\times 2L}$ with aid of the context matrices Y and Z collaboratively, which is shown in Figure 2. We utilize tucker decomposition to decompose $\mathcal{X}$ into the multiplication of a core tensor and three matrices as follows:

$$\mathcal{X} \approx O \times_W W \times_S S \times_T T \tag{2}$$

where $O \in \mathbb{R}^{d_W\times d_S\times d_T}$ is the core tensor and its entries show the level of interaction between the three components; $W \in \mathbb{R}^{N\times d_W}$, $S \in \mathbb{R}^{M\times d_S}$, and $T \in \mathbb{R}^{2L\times d_T}$ are the low rank latent factor matrices for workers, task categories and time slots; d_W, d_S, and d_T denote the dimensions of latent factors.

The two context matrices can be factorized in the same way. $Y \in \mathbb{R}^{2L\times M}$ can be factorized into the multiplication of two matrices, $Y = TS^T$, and $Z \in \mathbb{R}^{M\times Q}$ can be factorized into the multiplication of two matrices, $Z = SV$, where $V \in \mathbb{R}^{d_S\times Q}$ is the low rank latent factor for task features and Q denotes the dimension of task features. It is easy to see that tensor $\mathcal{X}$ shares matrix T with Y and shares matrix S with Z. Based on the knowledge of tensor $\mathcal{X}$ and two context matrices Y and Z, we then decompose $\mathcal{X}$, in which the loss function is defined in Equation 3 to control the errors.

$$\begin{aligned}\mathcal{L}(O,W,S,T,V) =& \frac{1}{2}||\mathcal{X} - O \times_W W \times_S S \times_T T||^2 + \\ & \frac{\lambda_1}{2}||Y - TS^T||^2 + \frac{\lambda_2}{2}||Z - SV||^2 + \\ & \frac{\lambda_3}{2}(||O||^2 + ||W||^2 + ||S||^2 + ||T||^2 + ||V||^2)\end{aligned} \tag{3}$$

where $||\cdot||$ denotes the Frobenius norm, $\frac{\lambda_3}{2}(||O||^2+||W||^2+||S||^2 + ||T||^2 + ||V||^2)$ is a regularization of penalties to avoid over-fitting, and λ_1, λ_2 and λ_3 are parameters controlling the contribution of different parts during the decomposition. Note that the whole missing values are regarded as zeros. In this work, we apply gradient descent algorithm to minimize the loss function, and then recover the missing values in $\mathcal{X}$ by multiplying the decomposed factors as $\mathcal{X}_{rec} = O \times_W W \times_S S \times_T T$.

Preference-aware Task Assignment

In the real-time scenario, where workers and tasks arrive dynamically and require immediate responses from an SC server, it is challenging to achieve the global optimal solution for PTA problem. Since an SC server only has a local knowledge of the available tasks and workers at any instance of time instead of a global view of all the workers and tasks, we will optimize the task assignment locally at every time instance by maximizing the current assignments and give higher priorities to workers who show more preference on the tasks simultaneously. In the sequel, we propose three heuristics to solve our proposed problem including *Basic*, *Spatial* and *Temporal* heuristics.

Preference-aware Task Assignment (PTA) Algorithm

Taking workers' preferences as the priority of task assignment, we propose a basic solution to solve the preference-aware task assignment problem by transforming it to Minimum Cost Maximum Flow (MCMF) problem.

The MCMF is based on a flow network graph representation of the task assignment problem for time instance t_i, in which the graph is represented by $G_i = (V, E)$ with V corresponding to the set of vertices and E the set of edges. Specifically, given a set of online workers, $W_i = \{w_1, w_2, ...\}$, and a set of available tasks, $S_i = \{s_1, s_2, ...\}$, at time instance t_i, the number of V and the number of E are fixed to $|W_i| + |S_i| + 2$ and $|W_i| + |S_i| + m$ respectively, where m is the number of available assignments for all the workers. The available assignments for worker w ($w \in W_i$) in time instance t_i, denoted as $\mathbb{A}_i^w$, should satisfy the following three conditions: $\forall < w, s >\in \mathbb{A}_i^w$, $s \in S_i$,

1) $d(w.l, s.l) \leq w.r$, and

2) $t_i + t(w.l, s.l) \leq s.p + s.\phi$ and

3) task s has not been performed by worker w,

where $d(w.l, s.l)$ is a given distance (e.g., Euclidean distance) between $w.l$ and $s.l$, and $t(w.l, s.l)$ is the travel time from $w.l$ to $s.l$. For the sake of simplicity, we assume all the workers share the same velocity, so the travel time cost between two locations can be estimated with their Euclidean distance, e.g., $t(w.l, s.l) = d(w.l, s.l)$. However, our proposed algorithms are not dependent on this assumption and can handle the case where workers are moving at different speeds. $|\mathbb{A}_i^w|$ denotes the number of available assignments for worker w and thus we can sum the number of available assignments for all the workers to get m, i.e., $m = \sum_{w\in W_i} |\mathbb{A}_i^w|$.

For the vertices construction, each worker w_j maps to a vertex, v_j, and each spatial task s_k maps to a vertex, $v_{|W_i|+k}$. In addition, two fictitious vertices src (labeled as v_0) and dst (labeled as $v_{|W_i|+|S_i|+1}$) are created to represent the source and destination respectively.

Figure 3 depicts an example of such network flow graph for three workers and six tasks at the same time instance. The corresponding edges are created using the following steps:

1) Edges associated from src to the vertices mapped from W_i are created. For each edge connecting src to v_j (mapped from w_j), denoted by (src, v_j), we set its capacity to 1 (i.e.,

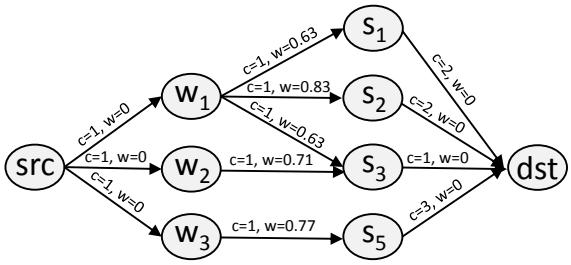


Figure 3: Flow Network-based Graph

$c(src, v_j) = 1$), since every worker is only capable of performing one task at the current time instance. The cost of the these edges are set to 0.

2) We generate $|S_i|$ edges connecting the vertices mapped from S_i to dst, where the capacity of each edge is set to $maxW$ since every task is to be assigned to at most $maxW$ workers. Same to edge (src, v_j), the cost of these edges are set to 0.

3) Due to the spatial and temporal constraints, we add an edge from v_j for every worker w_j to the vertex $v_{|W_i|+k}$ mapped from $s_k \in S_i$ if s_k can be assigned to w_j, i.e., $< w_j, s_k > \in \mathbb{A}_i^{w_j}$. For each edge $(v_j, v_{|W_i|+k})$, its capacity is set to one and its cost (denoted by $w(v_j, v_{|W_i|+k})$) can be measured as the ratio between 1 and worker's current preferences, i.e., $w(v_j, v_{|W_i|+k}) = \frac{1}{P_{w_j}^{\mathbb{T}}(s_k.c)+1}$.

The task assignment problem is now converted into a MCMF problem in the direct flow graph G_i from src to dst, which is to achieve the maximum flow of the graph while simultaneously minimize the cost. In our work, we use the Ford-Fulkerson algorithm (Kleinberg and Tardos 2005; Ford and Fulkerson 2009) to find the maximum flow of the network and then apply linear programming to minimize the cost of the flow (Kazemi and Shahabi 2012).

Spatial-weighted Preference-aware Task Assignment (SPTA) Algorithm

The PTA does not consider travel cost between worker and her designated tasks, which is critical in SC as workers have to physically go to the locations of the tasks in order to perform them. In our work, the travel cost between a worker w and a spatial task s, denoted as $d(w.l, s.l)$, is computed as a Euclidean distance between them. Due to the fact that a worker is more likely to accept nearby tasks (Ghinita, Ghinita, and Shahabi 2014), we give higher priorities to the closer tasks by verifying worker preferences at time instance t_i based on this heuristic. Given an online worker w and an available task s at time instance t_i, the weighted preference of worker w for task s, denoted by $P_w^{'}(s)$, can be computed as followed:

$$P_w^{'}(s) = P_w^{\mathbb{T}}(s.c) \cdot \delta(w.l, s.l) \quad (4)$$

$$\delta(w.l, s.l) = 1 - min(1, d(w.l, s.l)/w.r)$$

where $\delta(w.l, s.l)$ is a function calculating the discount to the worker's preference on the basis of her proximity to the task location. SPTA adapts PTA by calculating the weight of each edge $(v_j, v_{|W_i|+k})$ connecting w_j and s_k with the weighted preference, i.e., $w(v_j, v_{|W_i|+k}) = \frac{1}{P_w^{'}(s)+1}$.

Temporal-weighted Preference-aware Task Assignment (TPTA) Algorithm

This heuristic takes the temporal urgency of tasks into account to prioritize tasks, based on the intuition that a task which is further away from its deadline is more likely to be performed in the future, and vice versa. As a result, near-deadline tasks should have higher priorities to be assigned than others. Thus, in time instance t_i, we define the priority of a task s as the ratio between its remaining time and its vaild time, i.e., $\frac{s.p+s.\phi-t_i}{s.\phi}$ ($s.p \leq t_i$). TPTA modifies PTA through setting the weight for each edge $(v_{|W_i|+k}, v_{|W_i|+|S_i|+1})$ connecting s_k and dst with the priority of a task, i.e., $w(v_{|W_i|+k}, v_{|W_i|+|S_i|+1}) = \frac{s.p+s.\phi-t_i}{s.\phi}$ ($s.p \leq t_i$).

Experiment

Experimental Setup

We use a check-in dataset from Twitter to simulate our problem, which is a common practice in evaluation of SC platform (Deng, Shahabi, and Demiryurek 2013; Cheng et al. 2017; Dang, Nguyen, and To 2013). Since the original Twitter dataset does not contain category information of venues, we extract the category information associated with each venue from Foursquare with the aid of its API. The resulting dataset provides check-in data across USA except Hawaii and Alaska from September 2010 to January 2011, which includes locations of 62,462 venues and 61,412 users.

For our experiments, we assume the users are the workers of SC system since users who check in to different spots may be good candidates to perform spatial tasks in the vicinity of those spots, and their locations are those of the most recent check-in points. Moreover, we set the granularity of a time instance as 10 minutes (i.e., 10:00am-10:10am), during which the task requests and available workers will be packed and input to our framework. We assume all the users who check in during a time instance as online workers for that time instance. For each of the check-in venue, we use its location and the earliest check-in time of the day as the location and publish time of a task, respectively. Accordingly, the categories of check-ins are regarded as the categories of tasks and we extract 10 kinds of check-in features to simulate the task features. Checking in a spot is equivalent to accepting a task. In addition, we set $maxW$ for each task as the number of check-ins at the corresponding venue in a day. The travel cost is calculated by the Euclidian distance from the location of a worker to that of the task assigned to her. The default values of all parameters used in our experiments are summarized in Table 1. In the experiments of task assignment, we run the algorithms over four hours (i.e., 10:00am–2:00pm) of a day, and report the average results. All the algorithms are implemented on an Intel Core i5-2400 CPU @ 3.10G HZ with 8 GB RAM.

Table 1: Experiment Parameters

Parameter	Default value
Time span of historical data h	4 weeks
Valid time of tasks ϕ	1 h
Workers' reachable radius r	5 km
Number of tasks $\|S\|$	2000

Table 2: Performance of Different Methods for TPM

Methods	MAE	RMSE
AVF	0.2979	0.3384
TD	0.2671	0.3056
$TD + H$	0.2129	0.2406
$TD + H + Y + Z(HCTD)$	0.2054	0.2344

Experimental Results

Performance of Temporal Preferences Modeling. We first evaluate the performance of workers' temporal preferences modeling phase and its impact to subsequent task assignment. We set 1 hour as a time slot and use check-in data over a period of x weeks (and $x = 1, 2, 3, 4$ with a default value of 4) as historical data and check-in records of the day before as the recent data. The parameters (e.g., λ_1, λ_2 and λ_3) of loss function in tensor decomposition are set to 0.01.

To evaluate the accuracy of estimating workers' preferences for tasks, we adopt the widely-used measures, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). We randomly remove 20% of non-zero entries from the tensor $\mathcal{X}_r$, which are used as the testing set to evaluate the inferred values, and the remaining 80% are used as the training data. Then we introduce a baseline algorithm, Average Value Filling (AVF) algorithm, which complements a missing entry with the average of all non-zero entries in the tensor $\mathcal{X}$ that belong to the corresponding time slot. Moreover, we study the contribution of historical tensor (i.e., $\mathcal{X}_h$) and context matrices (i.e., Y and Z) for supplementing the missing entries. The methods are as followed:

1) AVF: Average value filling approach.

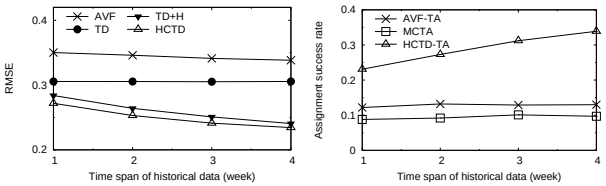
2) TD: Tensor decomposition approach that fills the missing entries by decomposing the tensor $\mathcal{X}_r$ solely based on its own non-zero entries.

3) TD+H: Tensor decomposition approach that fills the missing entries by decomposing the tensor with historical data (i.e., $\mathcal{X}_h$).

4) TD+H+Y+Z (HCTD): Tensor decomposition approach that fills the missing entries by decomposing the tensor with historical data, time and task context.

Table 2 shows the evaluation results, in which AVF achieves the worst performance while HCTD performs best followed by TD+H and TD. That demonstrates our method, HCTD, can provide more accurate estimates for worker preferences by considering historical data, temporal features and correlation between different task categories.

We further evaluate the performance of TPM phase and its impact to subsequent task assignment by varying the size of historical data. In particular, for accuracy of preference estimation, we compare the RMSE value of four different approaches including AVF, TD, TD+H, HCTD. In addition, we compare the assignment success rate of three different task assignment algorithms: HCTD-based Task Assignment (HCTD-TA) algorithm, AVF-based Task Assignment (AVF-TA) algorithm, and MCTA (Dang, Nguyen, and To 2013) that solves task assignment problem by transforming it into maximum flow problem without considering workers' preferences. When an SC server assigns a task s to worker w in a certain time instance (i.e., t_i), we consider the assignment successful for w if there exists a task sharing the same category with the assigned task s in the worker's task-performing list in the corresponding time slot $\mathbb{T}$ (i.e., $t_i \in \mathbb{T}$). Thus we introduce Assignment Success Rate (ASR), the ratio of successful assignments to the total assignments for all workers in a certain time instance, to measure the accuracy of task assignment.



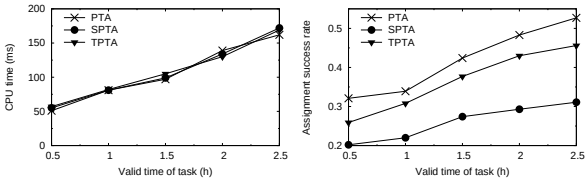
(a) RMSE (b) Assignment Success Rate

Figure 4: Performance of TPM: Effect of h

Effect of h. As shown in Figure 4a, naturally the accuracy of all algorithms except TD gradually increases as the time span of historical data grows. The estimation accuracy of TD is not affected by the historical data since it decomposes the tensor $\mathcal{X}_r$ solely without historical information. AVF achieves the worst performance amongst these methods. In addition, HCTD performs better than TD and TD+H, which testifies that the contributions of historical tensor and context matrices are effectiveness. In terms of task assignment success rate in Figure 4b, MCTA keeps constant since it does not consider worker preferences inferred from historical data. In addition, HCTD-TA has increasing assignment success rate with varying h due to its increasing estimation accuracy for workers' preferences, and it significantly outperforms the baseline algorithms for all values of h, which confirms the superiority of our proposed algorithm.

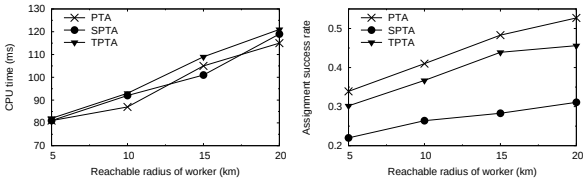
Performance of Preference-aware Task Assignment. Next we evaluate three different task assignment algorithms based on workers' temporal preferences generated by HCTD: PTA, SPTA and TPTA algorithm. Two metrics are compared among these three methods: 1) CPU cost: the CPU time cost for finding the task assignment in a time instance; 2) ASR: Assignment Success Rate.

Effect of ϕ. We first study the effect of the valid time ϕ of tasks. As illustrated in Figure 5a, all the methods have the similar performances with respect to CPU cost. This is because these methods all adopt the Maximum Flow Minimum Cost (MFMC) algorithm by just changing the weight



(a) CPU Cost (b) Assignment Success Rate

Figure 5: Performance of PTA: Effect of ϕ



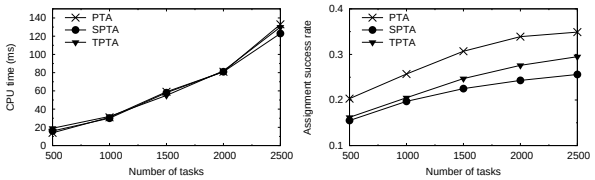
(a) CPU Cost (b) Assignment Success Rate

Figure 6: Performance of PTA: Effect of r

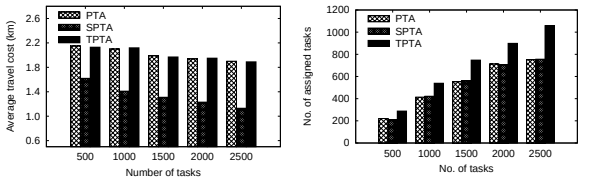
of the edges in the flow network graph, which does not affect the computation complexity. Another observation is that the CPU cost of all the methods increases almost linearly with ϕ, since the number of available tasks in a time instance grows when ϕ gets longer, which in turn leads to more edges in the flow network graph of MFMC to be searched. The ASR values of all methods are enhanced with the increasing ϕ (see Figure 5b) since a worker has more chance to be assigned her interested tasks when ϕ grows longer. SPTA and TPTA perform worse than PTA since they take workers' travel cost and tasks' expiration time into account respectively, which weakens the impact of workers' preferences on task assignment and leads to more inaccurate assignments.

Effect of r. We also study the effects of the length of workers' reachable radius r by changing it from 5 km to 20 km. From Figure 6a we can see that, the CPU cost of all the methods has similar increasing trend when r grows. The reason behind it is that all the methods apply MFMC algorithm and more workers with greater reachable radius tend to have more available task assignments, which leads to more edges in the flow network graph of MFMC. As shown in Figure 6b, the assignment success rate of the three approaches has a growing tendency as r is enlarged, with the similar reason of the effects of tasks' valid time, i.e., the larger the workers' reachable regions are, the more chance the SC server has to assign the workers their interested tasks.

Effect of $|S|$. To study the scalability of the proposed algorithms, we generate 5 datasets containing 500 to 2500 tasks by random selection from the original dataset in four hours (i.e., 10:00am–2:00pm) of a day. Besides the CPU cost and assignment success rate, we compare another two metrics among the three methods: 1) average travel cost of all the task assignments; 2) the total number of assigned tasks. As



(a) CPU Cost (b) Assignment Success Rate



(c) Average Travel Cost (d) Number of Assigned Tasks

Figure 7: Performance of PTA: Effect of $|S|$

expected, though the CPU cost increases as $|S|$ increases, our proposed algorithms perform well in improving the task assignment success rate, which is demonstrated in Figure 7a and Figure 7b. Figure 7c shows the average travel cost of all algorithms decreases since there is a higher probability that an assigned task is closer to a worker in a task-dense area. Furthermore, we notice that SPTA outperforms both PTA and TPTA by an astounding margin (up to 42%), which demonstrates the effectiveness of the spatial (travel cost) heuristic. As depicted in Figure 7d, naturally the assignments of all approaches increase when more tasks exist. The figure also illustrates the superiority of TPTA compared with PTA and SPTA in terms of the number of assigned tasks (up to 40%), which stems from applying the temporal heuristic. Moreover, the impact of temporal heuristic becomes more significant as the number of tasks grows. The reason behind it is that with a larger number of tasks, more tasks are soon to expire, and thus, prioritizing the near-deadline tasks to be assigned becomes more effective.

Conclusion

In this paper, we take an important step toward effective task assignment in spatial crowdsourcing based on workers' temporal preferences. We first address a few challenges arising from data sparsity and cold start by proposing a History-based Context-aware Tensor Decomposition (HCTD) method to model workers' temporal preferences for different task categories, and then design three different algorithms to find the optimal task assignment based on workers' temporal preferences in every time instance. To the best of our knowledge, it is the first work in spatial crowdsourcing that gives task assignment with workers' temporal preferences considered. Extensive empirical study demonstrates our proposed methods can significantly improve the effectiveness of task assignment.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (Grant No. 61502324, 61532018, 61836007, 61832017 and 61872258). This work is also partially supported by Alibaba Innovation Research (AIR).

References

Alt, F.; Shirazi, A. S.; Schmidt, A.; Kramer, U.; and Nawaz, Z. 2010. Location-based crowdsourcing:extending crowdsourcing to the real world. In *NordiCHI*, 13–22.

Ambati, V.; Vogel, S.; and Carbonell, J. 2011. Towards task recommendation in micro-task markets. In *AAAI*, 80–83.

Buchholz, S., and Latorre, J. 2011. Crowdsourcing preference tests, and how to detect cheating. In *ISCA*, 3053–3056.

Cheng, P.; Lian, X.; Chen, L.; and Han, J. 2015a. Task assignment on multi-skill oriented spatial crowdsourcing. *TKDE* 28(8):2201–2215.

Cheng, P.; Lian, X.; Chen, Z.; Fu, R.; Chen, L.; Han, J.; and Zhao, J. 2015b. Reliable diversity-based spatial crowdsourcing by moving workers. *VLDBJ* 8(10):1022–1033.

Cheng, P.; Lian, X.; Chen, L.; and Shahabi, C. 2017. Prediction-based task assignment in spatial crowdsourcing. In *ICDE*, 997–1008.

Dang, H.; Nguyen, T.; and To, H. 2013. Maximum complex task assignment: Towards tasks correlation in spatial crowdsourcing. In *ICPS*, 77–81.

Deng, D.; Shahabi, C.; and Demiryurek, U. 2013. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *SIGSPATIAL*, 324–333.

Deng, D.; Shahabi, C.; and Zhu, L. 2015. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *SIGSPATIAL*, 21.

Ford, L. R., J., and Fulkerson, D. R. 2009. Maximal flow through a network. *Canadian Journal of Mathematics* 8(3):399–404.

Ghinita, G.; Ghinita, G.; and Shahabi, C. 2014. A framework for protecting worker location privacy in spatial crowdsourcing. *VLDBJ* 919–930.

Kazemi, L., and Shahabi, C. 2012. Geocrowd: Enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*, 189–198.

Kleinberg, J., and Tardos, E. 2005. Algorithm design. *Prentice Hall*.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *Siam Review* 51(3):455–500.

Li, Y.; Yiu, M.; and Xu, W. 2015. Oriented online route recommendation for spatial crowdsourcing task workers. *SSTD* 137–156.

Song, T.; Tong, Y.; Wang, L.; She, J.; Yao, B.; Chen, L.; and Xu, K. 2017. Trichromatic online matching in real-time spatial crowdsourcing. In *ICDE*, 1009–1020.

Tong, Y.; She, J.; Ding, B.; Chen, L.; Wo, T.; and Xu, K. 2016a. Online minimum matching in real-time spatial data: Experiments and analysis. *VLDB* 9(12):1053–1064.

Tong, Y.; She, J.; Ding, B.; and Wang, L. 2016b. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE*, 49–60.

Tong, Y.; Wang, L.; Zhou, Z.; Ding, B.; Chen, L.; Ye, J.; and Xu, K. 2017. Flexible online task assignment in real-time spatial data. *VLDB* 10(11):1334–1345.

Tong, Y.; Wang, L.; Zhou, Z.; Chen, L.; Du, B.; and Ye, J. 2018a. Dynamic pricing in spatial crowdsourcing: A matching-based approach. In *SIGMOD*, 773–788.

Tong, Y.; Zeng, Y.; Zhou, Z.; Chen, L.; Ye, J.; and Xu, K. 2018b. A unified approach to route planning for shared mobility. *PVLDB* 11(11):1633–1646.

Yuen, M. C.; King, I.; and Leung, K. S. 2012. Task recommendation in crowdsourcing systems. In *CrowdKDD*, 22–26.

Zhao, Y.; Li, Y.; Wang, Y.; Su, H.; and Zheng, K. 2017. Destination-aware task assignment in spatial crowdsourcing. In *CIKM*, 297–306.