Preference-aware Task Assignment in Spatial Crowdsourcing: from Individuals to Groups

Zhao, Yan; Zheng, Kai; Yin, Hongzhi; Liu, Guanfeng ; Fang, Junhua; Zhou, Xiaofang

Published in:
Preference-aware Task Assignment in Spatial Crowdsourcing: from Individuals to Groups

DOI (link to publication from Publisher):
10.1109/TKDE.2020.3021028

Publication date:
2020

Preference-aware Task Assignment in Spatial Crowdsourcing: from Individuals to Groups

Yan Zhao, Kai Zheng*, *Senior Member, IEEE*, Hongzhi Yin, Guanfeng Liu, Junhua Fang, and Xiaofang Zhou, *Fellow, IEEE*

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 workers' 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 workers' preferences for different categories of tasks in different time slots. Several preference-aware individual 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. In order to make our proposed framework applicable to more scenarios, we further optimize the original framework by proposing strategies to allow each task to be assigned to a group of workers such that the task can be completed by these workers simultaneously, wherein workers' tolerable waiting time, consensus, and tasks' rewards are taken into consideration. We conduct extensive experiments using a real dataset, verifying the practicability of our proposed methods.

Index Terms—Spatial Crowdsourcing, Task Assignment, Worker Preference.

1 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 monitoring traffic condition, reporting local hot spot and moving a heavy stuff.

Most existing research focuses on the task assignment [5], [7], [12], [15], [31], [35], [36], [37], [38], [40], [43], [48], which aims to maximize the total number of completed tasks [23], the number of performed tasks for a worker with an optimal schedule [14], or the diversity score of assignments [8]. 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 he/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 the preference of a worker in his/her task-performing context. Among these existing contextual dimensions, time information, especially temporal dynamics, is of great importance since the characteristics of workers' 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 his/her lunch break but will definitely refuse to do it in his/her working hours. Therefore, incorporating temporal dynamics in workers' preferences can improve the effectiveness of spatial task assignment.

Several previous approaches infer workers' preferences from past task-performing patterns or explicit feedbacks [1], [2], [45]. 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 his/her long-term interest. But at any given time, a worker is also affected by his/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

- *Y. Zhao is with the Department of Computer Science, Aalborg University, Aalborg 9220, Denmark. Email: yanz@cs.aau.dk.*
- *K. Zheng is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China. Email: zhengkai@uestc.edu.cn. *K. Zheng is the corresponding author of the paper.*
- *H. Yin and X. Zhou are with the University of Queensland, Brisbane, Q4072, Australia. Email: zxf@itee.uq.edu.au, h.yin1@uq.edu.au.*
- *G. Liu is with the Department of Computing, Macquarie University, Sydney NSW2109, NSW Australia. Email: guanfeng.liu@mq.edu.au.*
- *J. Fang is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China. Email: jhfang@suda.edu.cn.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3021028, IEEE Transactions on Knowledge and Data Engineering

quality of task assignment in spatial crowdsourcing.

To address these challenges, we propose a Preference-aware Task Assignment (PTA) framework from individual aspect, 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 (HCT-D) with the aid of workers' task-performing history and two context matrices, we recover workers' 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.

Though our previous work [49] has already achieved the optimization goal of maximizing the total number of task assignments by taking personal preferences into account, it fails to consider group task assignment, which asks a group of workers with high consensus to perform a task (such as moving a heavy stuff) simultaneously.

Group activities including group task completion activities are essential ingredients in people's daily social life. The rapid development of SC services has greatly boosted group task completion activities by providing convenient platforms for workers to perform tasks together, in which the SC platforms need to target on not only individual workers but also worker groups. While making significant advances in the spatial crowdsourcing technology, most of the prior studies in this topic have focused on assigning tasks for individuals, which unfortunately cannot be effectively applied for group task assignment. Compared with individual task assignment, the major challenges in group task assignment are how to make a group of workers to reach the location of the assigned task at almost the same time (i.e., assigning a group of workers to each task without violating the tolerable waiting time constraint of each group member) and how to achieve high consensus among group members when assigning a group of workers to each task. To tackle these challenges, we develop an effective group task assignment framework considering both tolerable waiting time and consensus of group members. More specifically, we first utilize a fuzzy logic model to estimate the tolerable waiting time of workers for their reachable tasks. Then all the available worker groups are generated based on workers' tolerable waiting time constraint, and the group consensus can be calculated accordingly by taking group preference and similarity among group members into account. In the final assignment phase, we propose an algorithm based on tree decomposition to achieve the optimal task assignment.

As a summary, the major value-added extension over our preliminary work [49] is five folds.

1) We identify and study in depth a limitation in our previous individual PTA framework, which fails to be effectively applied for the group task assignment scenarios.

2) A tailor-made fuzzy logic model is designed to predict the tolerable waiting time of workers for their reachable tasks.

3) Based on workers' tolerable waiting time, we can generate the available worker groups and compute their consensus.

4) A tree-decomposition-based strategy is adopted to achieve the optimal task assignment with maximum group consensus.

5) We conduct extensive experiments with a real-world dataset, where the empirical results confirm that our solutions are effective in assigning spatial tasks to worker groups.

The remainder of this paper is organized as follows. Section 2 gives the preliminary concepts, along with a brief introduction of the framework overview. In Section 3, we introduce the workers' temporal preferences modeling approach. The proposed individual and group task assignment algorithms are presented in Section 4 and 5 respectively, followed by the experimental results in Section 6. Section 7 surveys the related work and Section 8 concludes this paper.

TABLE 1
Summary of Notations

Notation	Definition
s	Spatial task
$s.l$	Location of spatial task s
$s.p$	Publish time of spatial task s
$s.\phi$	Valid time of spatial task s
$s.c$	Category of spatial task s
$s.pt$	Processing time of spatial task s
$s.reward$	Reward of spatial task s
$s.maxW$	Maximum acceptable workers for s at certain time
$s.k$	The number of required workers for finishing s
w	Worker
$w.l$	Current location of worker w
$w.r$	Reachable radius of worker w
$w.off$	Offline time of worker w
S_w	Task-performing history for worker w
$\mathbb{T}$	Time slot
$P_w^{\mathbb{T}}(c)$	Preference of worker w for task category c in $\mathbb{T}$
A_i	Spatial task assignment instance set
X	A worker-task-time tensor
Y	A time-task matrix
Z	A task-feature matrix

2 Problem Statement

In this section, we briefly introduce a set of preliminary concepts, and then give an overview of our framework. Table 1 lists the major notations used throughout the paper.

2.1 Preliminary Concepts

Definition 1 (Spatial Task). A spatial task, denoted by $s =< s.l, s.p, s.\phi, s.c, s.pt, s.reward, s.maxW >$, 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 labelled with a category $s.c$ (e.g., taking photos, reporting local hot spot) and the processing time $s.pt$ required to finish task s. Each task also has a reward (specified by the task requester or SC platforms) and workers will receive a certain reward $s.reward$ for each task when they complete it. $s.maxW$ is the maximum number of workers allowed to be assigned to perform s at the same time instance in the individual task assignment.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3021028, IEEE Transactions on Knowledge and Data Engineering

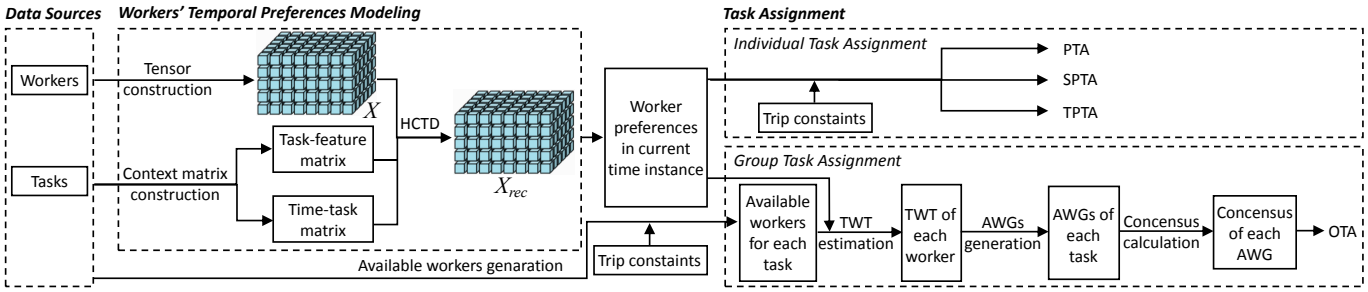


Fig. 1. PTA Framework

A spatial task s can be completed only when at least one worker can arrive at $s.l$ and finish it before its expiration time (i.e., $s.p + s.\phi$). Note that in the individual task assignment scenarios, each task s can be completed by several workers nearby during its valid time period while it is allowed to be assigned to up to $s.maxW$ worker at a certain time instance, which means task s can be completed by 1 to $s.maxW$ workers. $s.maxW$ can be specified by the task requester or SC platforms, which aims to avoid the misallocation of workers for tasks, i.e., most of the workers perform only a few tasks while leaving a large number of tasks unassigned. In group task assignment scenarios, each task s is associated with $s.k$ (the number of workers required to perform s) instead of $s.maxW$, and s needs to be completed by $s.k$ workers together.

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

In our model, a worker can handle only one task at a certain time instance, which is reasonable in practice.

Definition 3 (Task-performing History). Given a worker w who has performed n tasks in a time period, we define his/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 for task category c in a certain time slot $\mathbb{T}$, denoted by $P^{\mathbb{T}}_w(c)$, as the ratio of the number of tasks in category c to the number of total tasks that worker w has performed during $\mathbb{T}$, i.e.,

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

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

where $N^{\mathbb{T}}(S_w)$ is the number of tasks performed by w in 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, the goals of individual and group task assignment problems can be defined in the following.

1) **Individual task assignment problem** aims 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 for task categories and preferences for tasks' rewards at time instance t_i.

2) **Group task assignment problem** aims to assign each task to a group of workers to maximize the total consensus among group members by taking workers' tolerable waiting time, group preferences, group members' similarity and tasks' rewards into account at time instance t_i.

2.2 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) Task Assignment (TA) based on workers' temporal preferences.

The TPM procedure constructs a 3D tensor 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 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 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 X, which can effectively reduce decomposition error and improve inference accuracy.

The TA phase consists of individual and group task assignment process. In the individual task assignment part, based on the trip constraints including workers' reachable region and tasks' expiration time, we optimize the task assignment based on workers' current personal 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. In the group task assignment part, considering the trip constraints, we first obtain the available workers for each task. Then we estimate the Tolerable Waiting Time (TWT) of workers for their reachable tasks, based on which the Available Worker Groups (AWGs) are generated and the corresponding group consensus can be calculated. Finally, the Optimal Task Assignment (OTA) algorithm based on tree decomposition is employed to assign tasks.

3 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; 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.

3.1 Workers' Temporal Preferences Tensor Construction

In this section, we build a worker-task-time tensor, $X_r \in \mathbb{R}^{N\times M\times L}$, 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 $X_r(i,j,g)=e$ denotes the i-th worker's preference e on the j-th task category in time slot g (e.g., $10:00am-11:00am$). Obviously, there exists missing entries in tensor 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 X_r solely based on its own non-zero entries, so we introduce another tensor, 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 X_r shown in Figure 2. Clearly, X_h, representing the historical task-performing patterns and workers' long-term interests, is much denser than X_r. The error of supplementing X_r can be greatly reduced by decomposing X_r and X_h together.

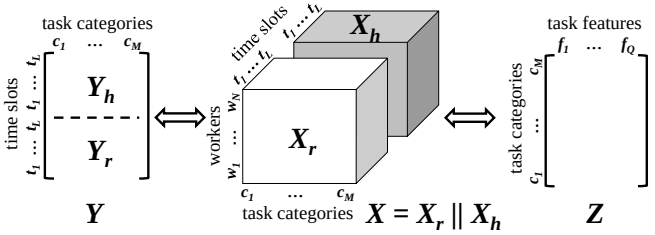


Fig. 2. Workers' Temporal Preferences Modeling

3.2 Context Matrix Construction

For more effective decomposition of the tensor X_r, we also construct another two matrices, i.e., time-task matrix Y and task-feature matrix Z.

3.2.1 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 $Y_r \in \mathbb{R}^{L\times M}$, $Y_r(g,j)$, represents the number of times that the tasks of category j have been performed in time slot g. 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 different time slots since these 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 he/she is likely to stay at his/her workplace and willing to perform some simple tasks, which do not affect his/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 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.

3.2.2 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.

3.3 Tensor Decomposition and Completion

Due to our goal of modeling workers' temporal preferences, we need to estimate the missing entries of X_r. A straightforward solution is leveraging tucker decomposition model, which is generally applied to higher-order principal component analysis [26]. 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 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 X_r are non-zero.

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

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

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 X shares matrix T with Y and shares matrix S with Z. Based on the knowledge of tensor X and two context matrices Y and Z, we then decompose X, in which the loss function is defined in Equation 3 to control the errors.

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

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 X by multiplying the decomposed factors as $X_{rec} = O \times_W W \times_S S \times_T T$.

4 Individual 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, the individual task assignment 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.

4.1 Preference-aware Task Assignment (PTA) Algorithm

Taking workers' preferences for task categories and tasks' rewards as the priority of task assignment, we propose a basic solution to solve the preference-aware task assignment problem by transforming it to a 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 conditions: $\forall < w, s > \in \mathbb{A}_i^w$, $s \in S_i$,

1) task s is located in the reachable circular range of worker w, i.e., $d(w.l, s.l) \leq w.r$, and

2) worker w can have enough time to finish task s before his/her offline time, i.e., $t_i + t(w.l, s.l) + s.pt \leq w.off$, and

3) worker w can have enough time to finish task s before s's expiration time, i.e., $t_i + t(w.l, s.l) + s.pt \leq s.p + s.\phi$ and

4) 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_g maps to a vertex, $v_{|W_i|+g}$. 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 four 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

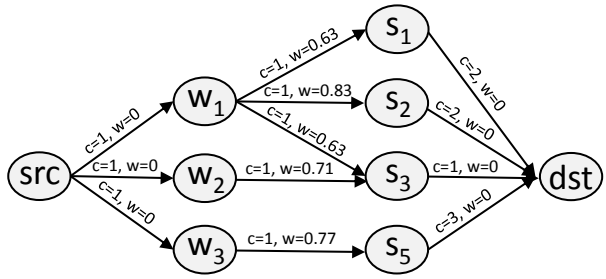


Fig. 3. Flow Network-based Graph

(mapped from w_j), denoted by (src, v_j), we set its capacity to 1 (i.e., $c(src, v_j) = 1$), 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 $v_{|W_i|+g}$ mapped from $s_g \in S_i$ if s_g can be assigned to w_j, i.e., $< w_j, s_g > \in \mathbb{A}_i^{w_j}$. For each edge $(v_j, v_{|W_i|+g})$, its capacity is set to one. The reward of a task (i.e., the reward obtained by a worker when he/she completes the task) reflects workers' preferences for it to some extend, i.e., workers tend to be more interested in performing the tasks with higher rewards. This is based on the intuition: reward (especially monetary reward) plays an important role to motivate workers to perform tasks in crowdsourcing systems [17], [21]. We normalize the reward values to lie between 0 and 1, using a Min-Max normalization procedure. The preference is frequency-based worker preference (see Definition 4), whose values are naturally between 0 and 1. To show both workers' preferences (for task categories) and their interests in the rewards received by completing tasks, we utilize a linear combination of preference and reward to represent the cost of an edge. More specifically, the cost (denoted by $w(v_j, v_{|W_i|+g})$) of each edge can be measured by the worker's current preference and the task's reward, i.e., $w(v_j, v_{|W_i|+g}) = \beta \frac{1}{P^{\mathbb{T}}_{w_j}(s_g.c)+1} + (1-\beta)\frac{1}{s_g.reward+1}$, where β ($0 \leq \beta \leq 1$) is the parameter controlling the contributions of the worker's current preference and the task's reward, $P^{\mathbb{T}}_{w_j}(s_g.c)$ ($t_i \in \mathbb{T}$) denotes worker w_j's current preference for task s_g and $s_g.reward$ denotes the reward of task s_g. The cost $w(v_j, v_{|W_i|+g})$ considers w_j's preferences on both the task category and task's reward.

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 minimizing the cost. In our work, we use the Ford-Fulkerson algorithm [25], [18] to find the maximum flow of the network and then apply linear programming to minimize the cost of the flow [23]. The Ford-Fulkerson algorithm is an iterative algorithm based on the idea of augmenting path. When the capacities are integers (the capacities of edges are set to 1 or $maxW$ in our work), the time complexity of Ford-Fulkerson is bounded by $O(|E| \cdot f_{max})$ [16], where $|E|$ is the number of edges and f_{max} is the maximum flow in the network.

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

The PTA does not consider the travel cost between workers and their 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 [19], 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:

$$P^{'}_{w}(s) = P^{\mathbb{T}}_{w}(s.c) \cdot \delta(w.l, s.l) \tag{4}$$
$$\delta(w.l, s.l) = 1 - min(1, d(w.l, s.l)/w.r)$$

where $\delta(w.l, s.l)$ is a function calculating the discount to the worker's preference on the basis of his/her proximity to the task location. SPTA adapts PTA by calculating the weight of each edge $(v_j, v_{|W_i|+g})$ connecting w_j and s_g with the weighted preference, i.e., $w(v_j, v_{|W_i|+g}) = \beta \frac{1}{P^{'}_{w}(s)+1} + (1-\beta)\frac{1}{s_g.reward+1}$ ($0 \leq \beta \leq 1$).

4.3 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., $\frac{s.p+s.\phi-s.pt-t_i}{s.\phi}$ ($s.p \leq t_i$), where $s.p$ is the publish time of task s, $s.\phi$ is the valid time of s, $s.pt$ is the processing time of s and t_i is the current time. TPTA modifies PTA through setting the weight for each edge $(v_{|W_i|+g}, v_{|W_i|+|S_i|+1})$ connecting s_g and dst with the priority of a task, i.e., $w(v_{|W_i|+g}, v_{|W_i|+|S_i|+1}) = \frac{s.p+s.\phi-s.pt-t_i}{s.\phi}$ ($s.p \leq t_i$).

5 Group Task Assignment

As the extension of our previous work [49], we will present in this section the algorithms that support group task assignment, where each task s has to be accomplished by $s.k$ ($s.k$, specified by the task requester or SC platform, is the number of workers required to perform s) workers together to guarantee its timely completion. Notice that the rationality behind our original work [49] that makes personalized task assignment for individuals is to discover workers' preference profiles in order to identify tasks that well match the profiles of targeted workers. However, for group task assignment, we argue that a good spatial crowdsourcing system not only needs to model workers' individual preferences but also considers the consensus among

group members, wherein the groups are often formed in ad-hoc manner (e.g., people temporarily gathering for a task). Moreover, the tolerable waiting time of each worker for their reachable tasks also plays a crucial role in affecting the quality of task assignment result since a worker is willing to wait others to perform the assigned task together only within his/her tolerable waiting time. Otherwise, he/she is highly likely to get away and abandon this task, which will leave the task unfinished. In this section, we optimize the group task assignment locally at every time instance by considering workers' tolerable waiting time and maximizing the total group consensus. In particular, we first calculate the available workers for each task by taking the trip constraints (i.e., workers' reachable region and tasks' expiration time) into consideration, and then estimate the Tolerable Waiting Time (TWT) of workers for their reachable tasks. Based on TWT, the Available Worker Groups (AWGs) are generated and the corresponding group consensus can be calculated. Lastly, we adopt a tree-decomposition-based technique [51] to achieve the optimal task assignment.

5.1 Available Workers Set Generation

Due to the constraint of tasks' expiration time as well as workers' available time and reachable range, each task can only be completed by a small subset of workers. Therefore, we firstly find the set of workers that can reach the location of each task without violating the constraints. The available worker subset for a task s, denoted as $s.AW$, should satisfy the following three conditions: $\forall w \in s.AW$,

1) worker w can have enough time to finish task s before the expiration time of s, i.e., $t_i + t(w.l, s.l) + s.pt \leq s.p + s.\phi$, and

2) worker w can have enough time to finish task s before his/her offline time, i.e., $t_i + t(w.l, s.l) + s.pt \leq w.off$, and

3) task s is located in the reachable range of worker w, i.e., $d(w.l, s.l) \leq w.r$,

where t_i denotes the current time instance, $t(a, b)$ is the travel time from location a to b, $s.pt$ is the processing time of task s, $s.p$ is the publish time of task s, $s.\phi$ is the valid time of task s, $w.off$ is the offline time of worker w, $d(a, b)$ is the travel distance from location a to b, and $w.r$ is the reachable distance of worker w. The above three conditions guarantee that a worker can travel from his/her origin to the location of task s (which is located in the reachable circular range of w) directly and complete task s before s expires as well as during w's available time (i.e., before w's offline time). If worker w is available for task s, i.e., $w \in s.AW$, we say s is a reachable task of w. It is easy to see the time complexity is $O(|W| \times |S|)$, where $|W|$ and $|S|$ are the numbers of workers and tasks respectively. As finding the available worker set for each worker is completely independent, it can be easily parallelized.

5.2 Tolerable Waiting Time Estimation

Estimating the Tolerable Waiting Time (TWT), during which a worker is willing to wait for other workers to perform the assigned task together and does not get away, is challenging in the field of spatial crowdsourcing. An intrinsically important issue in the estimation of TWT is dealing with the uncertainty since some information a worker or the SC system provides may be imprecise. For example, the SC system sometimes cannot obtain the precise location of a worker/task due to privacy issues or limitation of positioning technology. Moreover, workers' preferences on all types of tasks in the current time instance are estimated values instead of accurate values. Therefore, the estimation of TWT has the inherent ambiguity. Fuzzy logic, which is the logic underlying approximation or equivalently fuzzy reasoning, has evolved into a very useful tool for solving complex, real-world problems. The strength of fuzzy logic algorithms lies in their ability to systematically address the natural ambiguities in data measurement, prediction, and pattern recognition [11]. Since fuzzy logic is proved to be well suited to deal with uncertainties [28], [46], and there exists the inherent ambiguity in the estimation of TWT, we design a novel estimating algorithm of the tolerable waiting time by using fuzzy logic.

Intuitively, TWT is closely related to the location and time the worker requests. In particular, when a worker has enough available time, he/she tends to wait a long time for other collaborative workers. However, a worker may not be willing to spend a lot of time waiting if he/she has to travel a long distance from his/her location to the task's location. Furthermore, once a worker is interested in performing the task, he/she is willing to spend more time on this task and spends more time waiting other members to conduct this task together (i.e., his/her tolerable waiting time tends to be longer). Therefore, we take each worker's Urgency Degree (*UD*), Travel Cost (*TC*) and Preference (*P*) for a specified task as fuzzy variables, and obtain his/her corresponding TWT for the task.

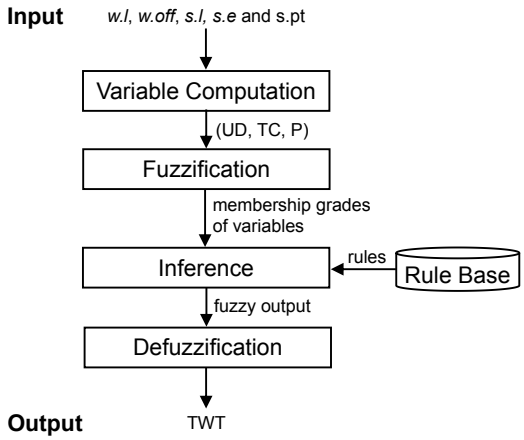
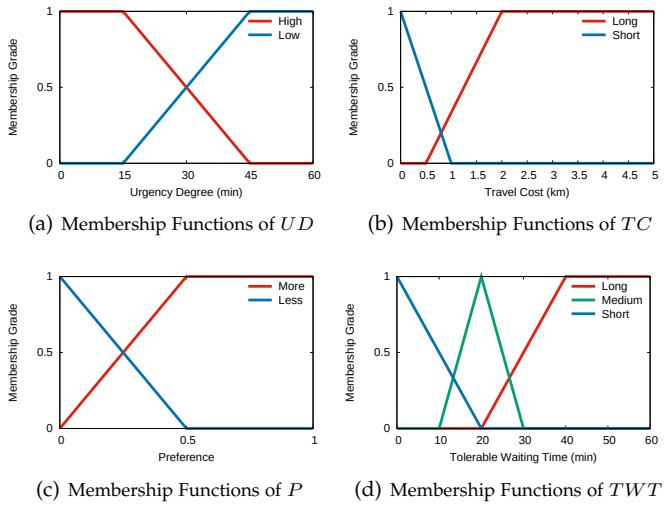


Fig. 4. Diagram of TWT Estimation

The overview of the algorithm for estimating the TWT of a worker for a task is illustrated in Figure 4. Taking a worker's location ($w.l$), his/her offline time ($w.off$), a task's location ($s.l$), its expiration time ($s.e$) and processing time ($s.pt$) as input, the algorithm first extracts the crisp values of fuzzy variables (including *UD*, *TC* and *P*) through the variable computation module, and then the fuzzy membership grades of fuzzy variables (as the linguistic values) are deduced from these crisp values based on the membership functions by fuzzification module. The reason we introduce the fuzzification module is that the rules offered by the rule base are presented as natural language with uncertainty,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3021028, IEEE Transactions on Knowledge and Data Engineering



(a) Membership Functions of UD (b) Membership Functions of TC

(c) Membership Functions of P (d) Membership Functions of TWT

Fig. 5. Membership Functions of Fuzzy Variables

which cannot be utilized without fuzzification process. In the subsequent step, the inference engine infers the fuzzy output from the previous fuzzification results of fuzzy variables according to the rules in the fuzzy rule base, where the rules can be formulated by human perception. This step aims to simulate the mode of thinking by human brain or human decision-making. Finally, the defuzzification module performs the inverse operation of fuzzification module to convert the fuzzy output into a crisp value as the final TWT output. In the following, we will elaborate the related technologies.

5.2.1 Variable Computation and Fuzzification

In this section, we introduce the computation and fuzzification process for variables (i.e., a worker's Urgency Degree (UD), Travel Cost (TC) and Preference (P)). Specifically, UD is determined by the worker's available time, the task's expiration time and processing time, which can be calculated by $UD = min\{w.off - t(w.l, s.l) - s.pt - t_i, s.p + s.\phi - t(w.l, s.l) - s.pt - t_i\}$ (where $w.off$ is the offline time of worker w, $t(w.l, s.l)$ is the travel time from $w.l$ to $s.l$, $s.pt$ is the processing time of task s, t_i is the current time, $s.p$ is the publish time of task s, and $s.\phi$ is the expiration time of task s). TC is equal to the travel distance between the the worker's location and task's location (i.e., $TC = d(w.l, s.l)$), and P refers to the worker's preference for the task, which has been obtained in Section 3.

Since a fuzzy set is usually approximated with a fuzzy set of triangle or trapezoidal type [34], [46], we adopt triangular and trapezoidal membership functions to model the relationship between $UD/TC/P$ and membership grades [20]. Based on the statistical data and empirical knowledge, we design the membership functions of both fuzzy variables and the output, which are illustrated in Figure 5. UD has two membership functions including *high* and *low* urgency degree functions (see Figure 5(a)). In Figure 5(b), TC also has two membership functions, i.e., *long* and *short* travel distance functions. From Figure 5(c) we can see the membership functions of P are *more* and *less* preference functions. For the output, TWT, it consists of three membership functions that are *long*, *medidum* and *short* tolerable waiting time functions, as depicted in Figure 5(d). The membership functions of TWT can be expressed in the following formulas and those of other variables can be represented in the same way, which are omitted due to space limit.

TABLE 2
Fuzzy Rule Base

Rule	UD	TC	P	TWT
R_1	high	short	more	short
R_2	high	short	less	short
R_3	high	long	more	short
R_4	high	long	less	medium
R_5	low	short	more	medium
R_6	low	short	less	medium
R_7	low	long	more	long
R_8	low	long	less	long

$$long\_TWT(x) = \begin{cases} 0, \text{if } 0 \le x \le 20 \\ (x-20)/20, \text{if } 20 < x < 40 \\ 1, \text{if } x \ge 40, \end{cases} \quad (5)$$

$$medium\_TWT(x) = \begin{cases} 0, \text{if } 0 \le x \le 10 \text{ or } x \ge 30 \\ (x-10)/10, \text{if } 10 < x < 20 \\ (30-x)/10, \text{if } 20 \le x < 30, \end{cases} \quad (6)$$

$$short\_TWT(x) = \begin{cases} (20-x)/20, \text{if } 0 \le x < 20 \\ 0, \text{if } x \ge 20, \end{cases} \quad (7)$$

where *long_TWT(x)*, *medium_TWT(x)* and *short_TWT(x)* denote the *long*, *medidum* and *short* tolerable waiting time functions respectively. x, the value of x-axis, is the tolerable waiting time.

5.2.2 Fuzzy Rule-based Inference

The fuzzy rule base we design is composed of eight rules, which is shown in Table 2. Each rule, represented as "if-then" format, provides the fuzzy output of TWT when the fuzzy variables take specific values. Taking rule R_2 as an example, R_2 means that if UD is high, TC is short and P is less, then TWT is short. This rule is intuitive since a worker is not willing to wait a long time for others to perform a task when he/she is very urgent and has less preference for this task.

According to the rules in the fuzzy rule base, the inference module can infer the fuzzy output from fuzzy variables. We adopt the MIN-MAX principle [27], [32] for rule matching and merging, wherein rule matching is performed by using fuzzy intersection (the min operator), while rule merging is conducted by employing fuzzy union (the max operator). For instance, if the membership grades of "UD is high", "TC is short" and "P is less" are a, b and c respectively, the inference strength of R_2, r_2, can be calculated by $r_2 = min\{a, b, c\}$ in the rule matching step. For rule merging process, since "TWT is short" is the consequence of R_1, R_2 and R_3, the membership grade of the fuzzy output, "TWT is short", is the maximal inference strength among R_1, R_2 and R_3, i.e., $o = max\{r_1, r_2, r_3\}$ (where o is the membership grade of the fuzzy output).

5.2.3 Defuzzification

In the final step, the crisp TWT can be derived from all the corresponding rules in the defuzzification module by

using a height defuzzification algorithm [33]. in which the computational formula is defined in Equation 8.

$$O = \frac{\sum_{i=1}^{n} o_i \times c_i}{\sum_{i=1}^{n} o_i}, \tag{8}$$

where o_i is the membership grade of the ith fuzzy output that can be obtained in the above step, c_i is the centroid of the ith membership function, and n is the number of membership functions of fuzzy output variables. As we can see from Equation 8, the final crisp output is the average of the centroids, which are weighted by the corresponding membership grades. Finally, we can estimate the tolerable waiting time (denoted as $w.TWT(s)$) of each worker w for each reachable task s.

Next, we will illustrate the process of TWT estimation. Suppose that the UD, TC and P of a worker for a task are 15, 1 and 0.5, respectively. Based on the membership functions in Figure 5, the membership grades of "UD is high" and "UD is low" are 1 and 0; the membership grades of "TC is long" and "TC is short" are 0.33 and 0; the membership grades of "P is more" and "P is less" are 1 and 0. According to the rules in the fuzzy rule base in Table 2, we can obtain the inference strength of R_3 as $r_3 = 0.33$ and those of the rest rules are 0 by the min operator, as a result of which the membership grades of "TWT is short", "TWT is medium" and "TWT is long" can be calculated as 0.33, 0 and 0 by the max operator. Then according to Equation 8, we can get that the TWT is 6.67 as the centroids of *long TWT(x)*, *medium TWT(x)* and *short TWT(x)* are 6.67, 20 and 46.67 respectively.

5.3 Available Worker Group Generation and Consensus Calculation

5.3.1 Available Worker Group Generation

Given the available workers for each task s and their tolerable waiting time for s, we next find the set of Available Worker Groups (AWGs), each of which has $s.k$ workers, denoted as $s.k$-AWG. Each $s.k$-AWG should satisfy the following condition: $\forall w_i, w_j \in s.k$-AWG, $t(w_i.l, s.l) + w_i.TWT(s) \geq t(w_j.l, s.l)$, where $t(a, b)$ denotes the travel time between location a and b, and $w_i.TWT(s)$ denotes the tolerable waiting time of worker w_i for task s that has been calculated in Section 5.2. The time complexity of computing AWG is $O(|S| \cdot |AW_{max}|^k)$, where $|S|$ is the number of tasks to be assigned, $|AW_{max}|$ denotes the maximal number of available workers for all the tasks, and k is the number of group members.

5.3.2 Consensus Calculation

The goal of group task assignment is to assign each task to a group of workers that reflects the interests and preferences of all the group members. In general, group members may not always have the same tastes and the group task assignment process should manage the heterogeneity of groups. As a consequence, a consensus score for each group needs to be carefully designed. The consensus process is necessary to solve group task assignment, which is used to obtain a final solution with the certain level of agreement among the workers. Intuitively, there are two main aspects to the consensus score. First, the score needs to reflect the degree to which the task is preferred by all the members. The more group members prefer a task, the higher its score should be for the group. Second, the score needs to reflect the degree of similarity among members since similar workers are more willing to collaborate with each other.

Getting the set of available worker groups for task s, we next calculate the consensus score for each group, which consists of two components: *group preference* and *group members' similarity*. More specifically, since the individual preferences are given (in Section 3), we simply adopt the average aggregation strategy, the most prevalent mechanism being employed currently [22], to get the group preference. Group members' similarity is evaluated through a standard metric (i.e., cosine similarity) based on the preference ratings matrix, which measures how similar the preferences of the group members are. The consensus score for a group G, denoted by $con(G)$, can be calculated as follows:

$$con(G) = \alpha \mathbb{P}(G) + (1 - \alpha) sim(G), \tag{9}$$

$$\mathbb{P}(G) = \frac{1}{|G|} \sum_{w \in G} (P_w^{\mathbb{T}}(s.c)), \tag{10}$$

$$sim(G) = \frac{1}{|G|(|G| - 1)} \sum_{w_i, w_j \in G} \frac{v_{w_i} \cdot v_{w_j}}{||v_{w_i}|| \times ||v_{w_j}||}, \tag{11}$$

where α is a parameter controlling the contributions of group preference and group members' similarity, $\mathbb{P}(G)$ denotes the group preference for G, $|G|$ is the number of group members, $P_w^{\mathbb{T}}(s.c)$ is the preference of worker w in task category $s.c$ in the current time slot $\mathbb{T}$, and v_w is the vector of the preference ratings expressed by worker w for all the categories of tasks. $\frac{v_{w_i} \cdot v_{w_j}}{||v_{w_i}|| \times ||v_{w_j}||}$ denotes the cosine similarity between v_{w_i} and v_{w_j}.

5.4 Group Task Assignment

The group task assignment aims to maximize the overall consensus of worker groups, in which we give more priority to the task with a higher reward.

Lemma 1. The Group Task Assignment problem is NP-hard.

Proof 1. We prove Lemma 1 through a reduction from the 0-1 knapsack problem. A 0-1 knapsack problem can be described as follows. Given a set O of n items, in which each item o_i is labelled with a weight k_i and a value v_i, the 0-1 knapsack problem is to find a subset O' of O that maximizes $\sum_{o_i \in O'} v_i$ subjected to $\sum_{o_i \in O'} k_i \leq M$, where M is the maximum weight capacity.
For a given 0-1 knapsack problem, we can transform it to an instance of the Group Task Assignment problem as follows. We give a spatial task set S of n tasks and a worker set W of M workers. Each task s_i in S needs k_i workers to complete and obtains the corresponding worker group consensus, v_i. Given this mapping, we can show that the 0-1 knapsack problem can be solved, if and only if the transformed Group Task Assignment problem can be solved. Since 0-1 knapsack problem is known to be NP-hard [42], Group Task Assignment problem is also NP-hard.

Intuitively, we can apply a simple greedy algorithm to find the available worker group with maximum consensus

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3021028, IEEE Transactions on Knowledge and Data Engineering

for each task as the assignment result. However, this is hardly a practical solution.

Since the global result is the union of one possible Available Worker Group (AWG) of all tasks, we adopt an Optimal Task Assignment (OTA) algorithm with tree-decomposition-based strategy [48], [51] to find the optimal task assignment result with maximal consensus, in which we give higher priority to the tasks with higher rewards. The OTA algorithm consists of the following steps:

1) We first construct a task dependency graph, $G(V, E)$, according to the dependency relationship among tasks (two tasks are dependent with each other if they share the common available workers; otherwise they are independent), where each vertex $v \in V$ represents a task $s_v \in S$. An edge $e(u, v) \in E$ exists between u and v if the two tasks s_u and s_v are dependent with each other. The time complexity of the task dependency graph construction is $O(|S|^2 \cdot |AW_{max}|)$, where $|AW_{max}|$ is the maximal number of available workers for all the tasks.

2) We utilize the tree-decomposition strategy to separate all the tasks into clusters (each cluster is a maximal clique of the task dependency graph) and organize them into a balance tree by Recursive Tree Construction (RTC) algorithm [48], [51], such that the tasks in sibling nodes of the tree do not share the same available workers. The time complexity of this step is $O(\sum_i^m(|\mathbb{X}^i| + |G^i_{sub}| \cdot (|V^i| + |E'^i|)))$, where m is the number of recursions of the RTC algorithm. $\mathbb{X}^i$, G^i_{sub}, V^i and E'^i denote the task cluster set, subgraph set, vertex set and edge set to be checked in the ith recursion.

3) Finally, the tree can be traversed in a depth-first manner to find the optimal assignment. When encountering the tasks that can generate the same worker group consensus, the task with a higher reward is given the priority to be assigned. The time complexity of this search procedure is $O(\sum_i^r(|S^i_N| \cdot |Q^i_s| + |N^i_{child}|))$, where r denotes the number of recursions when searching. $|S^i_N|$ is the number of tasks in the tree node N in the ith recursion, $|Q^i_s|$ is the number of $s.k$-AWGs (each $s.k$-AWG is an available worker group containing $s.k$ workers, see Section 5.3.1) for task s in the ith recursion, and $|N^i_{child}|$ is the number of child nodes of N in the ith recursion.

6 Experiment

6.1 Experimental Setup

Due to the lack of real worker and task data, we use a check-in dataset from Twitter to simulate our problem, which is a common practice in evaluation of SC platform [7], [13], [14]. 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. The offline time of workers are generated following the uniform distribution. 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. For each task category, we set its processing time as the average time of workers staying in this task (i.e., the corresponding spot), and the tasks with the same category have the same processing time. The reward of each task is proportional to its processing time, and we simply set the reward of each task as its processing time. Checking in a spot is equivalent to accepting a task. For individual task assignment, we set $maxW$ for each task as the number of check-ins at the corresponding venue in a day. For the sake of simplicity, we assume all the workers share the same velocity, which is set as 5km/h. The default values of all parameters used in our experiments are summarized in Table 3. In the experiments of task assignment, we run the algorithms over 4 hours (i.e., 10:00am–2:00pm) of a day, and report the average results. The parameter α (controlling the contributions of group preference and group members similarity) is set to 0.5, and the parameter β (controlling the contribution of the worker's current preference and task's reward) during task assignment is also set to 0.5.

TABLE 3
Experiment Parameters

Parameters	Values
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
Number of workers, $\|W\|$	2000
Number of workers allowed to be assigned to a task in group task assignment, k	2

As Twitter does not contain explicit group information, we extract implicit group task completion activities as follows: we assume if a set of users visit the same spot or different spots with the same category which are near to each other (e.g., the distance between any two spots is less than 5km in our experiments) in one hour, they are the members of a group and the corresponding activities are group task completion activities. All the algorithms are implemented on an Intel Core i5-2400 CPU @ 3.10G HZ with 8 GB RAM.

6.2 Experimental Results

6.2.1 Performance of Temporal Preferences Modeling

We first evaluate the performance of workers' temporal preferences modeling phase and its impact to the subsequent individual 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, which means the contributions of time-task

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3021028, IEEE Transactions on Knowledge and Data Engineering

TABLE 4
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**

matrix, task-feature matrix and regularization of penalties for the tensor decomposition are same.

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 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 X that belong to the corresponding time slot. Moreover, we study the contribution of historical tensor (i.e., 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.

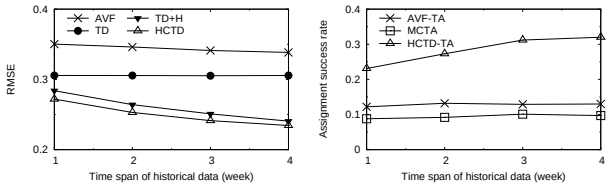
2) TD: Tensor decomposition approach that fills the missing entries by decomposing the tensor X_r solely based on its own non-zero entries.

3) TD+H: Tensor decomposition approach that fills the missing entries by decomposing the tensor with historical data (i.e., 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.

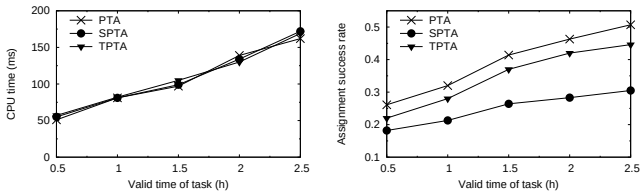
Table 4 shows the evaluation results, in which AVF achieves the worst performance while HCTD performs best followed by TD+H and TD. That demonstrates our method, HCTD, can provide more accurate estimates for worker preferences by considering historical data, temporal features and correlation between different task categories.

We further evaluate the performance of TPM phase and its impact to the subsequent individual 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 [13] 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

Fig. 6. Performance of TPM: Effect of h



(a) CPU Cost (b) Assignment Success Rate

Fig. 7. Performance of Individual Task Assignment: Effect of ϕ

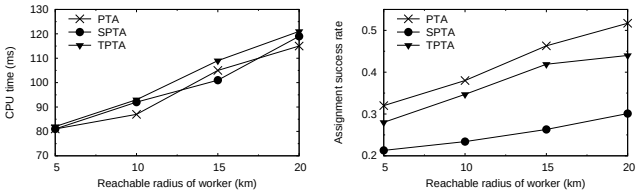
Effect of h. As shown in Figure 6(a), naturally the accuracy of all algorithms except TD gradually increases as the time span of historical data grows. The estimation accuracy of TD is not affected by the historical data since it decomposes the tensor X_r solely without historical information. AVF achieves 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 6(b), 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.

6.2.2 Performance of Individual Task Assignment

Next we evaluate three different individual 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.

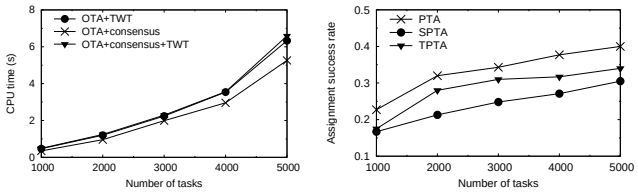
Effect of ϕ. We first study the effect of the valid time ϕ of tasks. As illustrated in Figure 7(a), all the methods have the similar performances with respect to CPU cost. This is because these methods all adopt the Maximum Flow Minimum Cost (MFMC) algorithm by just changing the weight of the edges in 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 7(b)) since a worker has more chance to be assigned his/her interested tasks when ϕ grows

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3021028, IEEE Transactions on Knowledge and Data Engineering

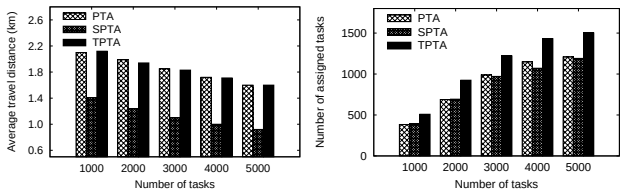


(a) CPU Cost (b) Assignment Success Rate

Fig. 8. Performance of Individual Task Assignment: Effect of r



(a) CPU Cost (b) Assignment Success Rate



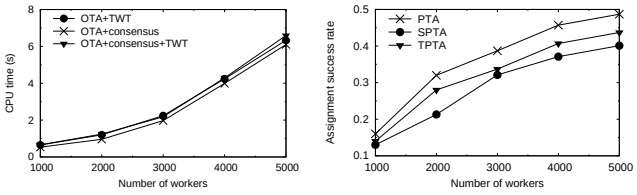
(c) Average Travel Distance (d) Number of Assigned Tasks

Fig. 9. Performance of Individual Task Assignment: Effect of $|S|$

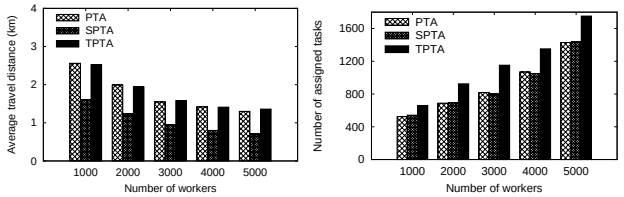
longer. SPTA and TPTA perform worse than PTA since they take workers' travel distance and tasks' expiration time into account respectively, which weakens the impact of workers' preferences on task assignment and leads to more inaccurate assignments.

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 8(a) 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 8(b), 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 workers their interested tasks.

Effect of $|S|$. To study the scalability of the proposed algorithms, we generate 5 datasets containing 1000 to 5000 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 distance of all the task assignments; 2) the total number of assigned tasks. As 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 9(a) and Figure 9(b). Figure 9(c) shows the average



(a) CPU Cost (b) Assignment Success Rate



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

Fig. 10. Performance of Individual Task Assignment: Effect of $|W|$

travel distance 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 44.5%), which demonstrates the effectiveness of the spatial (travel distance) heuristic. As depicted in Figure 9(d), 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 34.3%), 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.

Effect of $|W|$. We also investigate how the number of workers affects the efficiency and effectiveness of task assignment. We can see from Figure 10(a), the running time of all methods increases for larger number of workers, since there are more worker-and-task assignments to process. As expected, the assignment success rates for all the algorithms gradually increase as $|W|$ grows, which is indicated in Figure 10(b). PTA performs best since it only consider workers' preferences while ignoring other spatio-temporal constraints (i.e., workers' travel distance and tasks' expiration time). From Figure 10(c) we can see that the average travel distance shows a downward trend with the varying $|W|$ since an assigned task is more likely to be closer to a worker when there are more workers. Figure 10(d) depicts that the number of assigned tasks increases as the number of workers grows. Moreover, similar to the previous experiments, TPTA is the superior approach in terms of improving the number of task assignments (up to 43.8% better than others).

6.2.3 Performance of Group Task Assignment

We evaluate the performance of group task assignment by the following algorithms:

1) *OTA+TWT*: the Optimal Task Assignment algorithm considering workers' tolerable waiting time and their average preference while ignoring the group similarity.

2) *OTA+concensus*: the Optimal Task Assignment algorithm based on group consensus while ignoring workers' tolerable waiting time.

3) *OTA+concensus+TWT*: the Optimal Task Assignment algorithm taking both workers' tolerable waiting time and group consensus into account.

Five metrics are compared among the above algorithms:

1) CPU time: the CPU time cost for finding a task assignment in a time instance.

2) Total consensus.

3) ASR: Assignment Success Rate that is the ratio of successful assignments to the total assignments for all workers in a certain time instance. Note that once all the group members actually perform (check in) the tasks (spots) with the same category which are near to each other (e.g., the distance between the tasks is less than 5km in our experiments) in one hour, we regard this task assignment as a successful assignment.

4) Number of assigned tasks.

5) Average waiting time: the average waiting time per worker for performing a task.

Effect of k. We first investigate how k, the required number of workers allowed to perform a task simultaneously, affects the task assignment results. Figure 11(a) depicts the running time of different approaches. As expected, larger k means on average each task has less available worker groups, which results in less running time during the process of group task assignment. $OTA + TWT$ and $OTA + TWT + consensus$ run slower than $OTA + consensus$ as they have to compute the tolerable waiting time of each worker for his/her reachable task. Obviously, the CPU time of $OTA + TWT + consensus$ is slightly higher than that of $OTA + TWT$ due to the calculation of worker groups' consensus. As illustrated in Figure 11(b), though the total consensus of the three methods declines with k, $OTA + consensus$ and $OTA + TWT + consensus$ achieve more overall consensus than $OTA + TWT$, which in turn lead to the higher assignment success rate than $OTA + TWT$ as confirmed in Figure 11(c), demonstrating the benefit of considering group consensus in task assignment. For the number of assigned tasks in Figure 11(d), $OTA+consensus$ performs best regardless of k since ignoring the constraint of workers' tolerable waiting time means a task has more chance to be assigned to a worker group. The average waiting time of workers gradually increases with k (see Figure 11(e)) since when more members are involved in a worker group (i.e., when k gets larger), a worker has to wait for more other workers to complete a task together. This will result in more waiting time for this worker. Moreover, $OTA + TWT$ and $OTA + TWT + consensus$ have less average waiting time than $OTA + consensus$, confirming the advantage of considering workers' tolerable waiting time during task assignment.

Effect of $|S|$. In this set of experiments, we evaluate the scalability of all the approaches by varying the number $|S|$ of tasks from 1000 to 5000. Figure 12 shows the effect of $|S|$. As illustrated in Figure 12(a), the running time of all methods increases with the rising number of tasks, while the growth of computational cost for $OTA + TWT$ and $OTA + TWT + consensus$ is relatively faster due to the tolerable waiting time calculation of workers for each task in these two methods. $OTA + consensus$ achieves the highest total consensus and the best assignment success rate among the three methods (depicted in Figure 12(b)and 12(c)), since it does not consider workers' tolerable waiting time. Taking workers' tolerable waiting time into account will lead to smaller number of available worker groups for each task, which influences the overall consensus of workers and the success of task assignment. From Figure 12(d) we can see $OTA + consensus$ can assign more tasks since it ignores workers' tolerable waiting time. Besides, $OTA + TWT$ and $OTA + TWT + consensus$ have the similar performance in terms of the number of assigned tasks. The average waiting time of all the approaches is not associated with the varying $|S|$, as described in Figure 12(e). The reason behind it is that we only consider worker groups' consensus when assigning available worker groups to tasks in the Optimal Task Assignment (OTA) algorithm in Section 5.4. Besides, $OTA + consensus$ performs worse than $OTA + TWT$ and $OTA + TWT + consensus$ in terms of the average waiting time, demonstrating the advantage of TWT optimization.

Effect of $|W|$. In our final set of experiments, we measure the performance of our approaches with respect to expanding the number of workers ($|W|$) from 1000 to 5000. As Figure 13(a) shows, with a larger $|W|$, the CPU time increases. The main reason behind it is that the number of available workers to be assigned grows when $|W|$ gets larger, which in turn leads to longer time cost. For total consensus in Figure 13(b) and assignment success rate in Figure 13(c), $OTA + consensus$ performs best as it focuses on the consensus of worker group without considering the constraint of workers' tolerable waiting time. In Figure 13(d), the number of assigned tasks increases due to the fact that more spatial tasks can be conducted by more workers. Figure 13(e) illustrates the average waiting time fluctuates as $|W|$ changes with the similar reason of the effects of $|S|$, i.e., the OTA algorithm only focuses on workers' consensus when assigning each task the suitable available worker group. Both $OTA + TWT$ and $OTA + TWT + consensus$ outperform $OTA + consensus$ in average waiting time, which demonstrates the critical impact of workers' TWT heuristic.

Summary: The take-away message of our empirical study on group task assignment can be summarized as follows:

1) $OTA + consensus$ algorithm performs better than others in total consensus, task assignment success rate and number of assigned tasks.

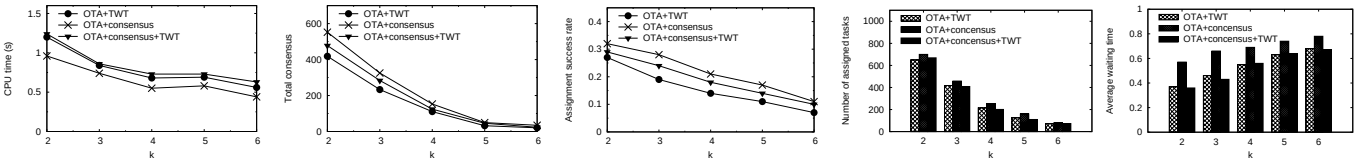
2) Although the total consensus of $OTA + TWT$ is less than others, it can effectively reduce the average waiting time.

3) $OTA + consensus + TWT$ achieves good balance among total consensus, task assignment success rate, number of assigned tasks and average waiting time.

7 Related Work

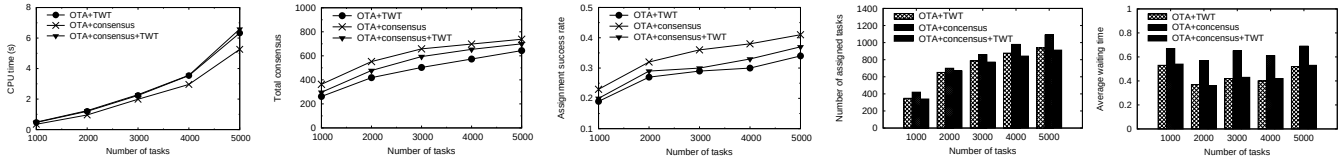
Spatial Crowdsourcing (SC) is a type of online crowdsourcing, which employs smart device carriers as workers to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3021028, IEEE Transactions on Knowledge and Data Engineering



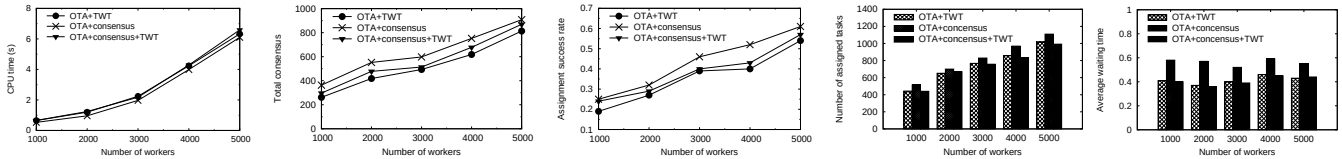
(a) CPU Cost (b) Total Consensus (c) Assignment Success Rate (d) Number of Assigned Tasks (e) Average Waiting Time

Fig. 11. Performance of Group Task Assignment: Effect of k



(a) CPU Cost (b) Total Consensus (c) Assignment Success Rate (d) Number of Assigned Tasks (e) Average Waiting Time

Fig. 12. Performance of Group Task Assignment: Effect of $|S|$



(a) CPU Cost (b) Total Consensus (c) Assignment Success Rate (d) Number of Assigned Tasks (e) Average Waiting Time

Fig. 13. Performance of Group Task Assignment: Effect of $|W|$

physically move to some specified locations and perform spatial tasks [41], [50], [29]. Based on the task publish mode, SC can be classified into Server Assigned Tasks (SAT) mode and Worker Selected Tasks (WST) mode [23]. In SAT mode, the server assigns each task to nearby workers based on the system optimization goals such as maximizing the number of assigned tasks after collecting all the locations of workers [15], [23], [24]. Kazemi and Shahabi [23], for example, formulate SC as a task-matching problem between workers and tasks. They aim to maximize the total number of assigned tasks while conforming to workers' constraints, with the assumption that the server has a global knowledge of tasks and workers at every time instance. [47] focuses on the latency-oriented task completion problem, which explores the trade-off between latency and quality for both offline and online task assignment in spatial crowdsourcing. [39] proposes a Global Online Micro-task Allocation (GOMA) problem and designs a two-phase based framework under the random order model. Based on this framework, several Two-phase-based Global Online Allocation (TGOA) algorithms are developed to effectively and efficiently solve the GOMA problem. While in WST mode, the server publishes various spatial tasks online, and workers can select any tasks without the coordination with the server [14], [15]. For instance, Deng et al. [14] formulate SC as a scheduling problem by reducing it into a specialized *Traveling Salesman Problem*. The authors propose exact and approximation algorithms to find a schedule which maximizes the number of tasks that can be completed by a worker where both travel cost of workers and expiration time of tasks have been taken into consideration.

Quality assurance is a major challenge among these spatial task assignment work. Workers tend to complete the assigned tasks with poor quality if there is not any quality control strategy (e.g., assigning tasks to workers based on their preferences). However, the existing studies mentioned above do not take workers' preferences for tasks into account or just consider workers' preferences based on their travel cost (i.e., workers are more likely to accept nearby tasks [19]). Although a few researches explore workers' preferences for tasks in crowdsourcing [1], [2], [45], they just infer workers' preferences from past task-performing patterns or explicit feedbacks without considering workers' temporal preferences for different tasks. In other research areas, e.g., personalized product recommendation, some work uses new techniques of machine learning to learn users' preferences [3]. For example, Quadrana et al. [30] introduce the usage of Gated Recurrent Units (GRUs) with the collaborative filtering method for learning users' preferences, where GRU is a new generation of Recurrent Neural Network (RNN) and regarded as a variant of Long Short-Term Memory (LSTM) network [9]. GRUs can reduce the computational burden and perform as good as LSTM [10]. [3] learns a user's preferences from his/her ongoing session (such as the user's behaviors, meta data of the products, etc.) by using the Gated Recurrent Units (GRUs) with attention function. The above GRU models aim to estimate users' preferences by finding the sequential correlations

among users' behaviors (e.g., users' click/view/buy sequence). However, the task-performing behaviors of workers in spatial crowdsourcing may not be in sequence. For example, a worker can only perform a task during his/her lunch break or on his/her way home. [44] develops an efficient Graph Convolutional Neural Networks (GCNN) model for web-scale recommender systems, which can learn embeddings (representing users' preferences) for nodes (i.e., items) in web-scale graphs for item-item recommendation (e.g., related-pin recommendation in Pinterest [1]). In spatial crowdsourcing, the task assignment problem can be regarded as a worker-task matching problem in a bipartite network or a flow network [15], [37], which is a Heterogeneous Information Network (HIN) since its nodes and edges belong to multiple types. The GCNN model has certain weaknesses when being applied to our problem settings since the random walk it adopts does not consider the heterogeneity of a HIN.

While making significant advances in the spatial crowdsourcing technology, most of the prior research studies in this topic have focused on assigning tasks for individuals, which unfortunately cannot be effectively applied for group task assignment.

In recent years, many efforts have been devoted to improve the group task assignment, also called collaborative task assignment. Cheng et al. [4] propose a task assignment problem in spatial crowdsourcing, called Cooperation-Aware Spatial Crowdsourcing (CA-SC), in which they design both task-priority greedy approach and game theoretic approach to solve the CA-SC problem, achieving high total cooperation quality scores. By developing several effective heuristic approaches, [6] assigns a task to multiple workers with different skills such that the skills between workers and tasks match with each other, and workers' benefits are maximized under the budget constraint. However, the previous studies do not consider the tolerable waiting time of workers during task assignment and ignore the consensus among group members, which have great impact on the success of group task assignment.

8 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 (HCT-D) 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 solve the individual task assignment problem. We further consider another task assignment scenario where a task needs be completed by a group of workers simultaneously, and propose an effective group task assignment solution by taking tolerable waiting time and consensus of group members, as well as tasks' rewards into consideration. Extensive empirical study demonstrates the effectiveness of our proposed methods.

1. http://pinterest.com/

Acknowledgment

This work is partially supported by Natural Science Foundation of China (No. 61972069, 61532018, 61836007 and 61832017), Sichuan Science and Technology Program under Grant 2020JDTD0007 and Alibaba Innovation Research (AIR).

References

[1] V. Ambati, S. Vogel, and J. Carbonell. Towards task recommendation in micro-task markets. In *AAAI*, pages 80–83, 2011.
[2] S. Buchholz and J. Latorre. Crowdsourcing preference tests, and how to detect cheating. In *ISCA*, pages 3053–3056, 2011.
[3] J. Chen and A. Abdul. A session-based customer preference learning method by using the gated recurrent units with attention function. *Access*, 7:17750–17759, 2019.
[4] P. Cheng, L. Chen, and J. Ye. Cooperation-aware task assignment in spatial crowdsourcing. In *ICDE*, pages 1442–1453, 2019.
[5] P. Cheng, X. Lian, L. Chen, and J. Han. Task assignment on multi-skill oriented spatial crowdsourcing. *TKDE*, 28(8):2201–2215, 2015.
[6] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao. Task assignment on multi-skill oriented spatial crowdsourcing. *TKDE*, 28(8):2201–2215, 2016.
[7] P. Cheng, X. Lian, L. Chen, and C. Shahabi. Prediction-based task assignment in spatial crowdsourcing. In *ICDE*, pages 997–1008, 2017.
[8] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao. Reliable diversity-based spatial crowdsourcing by moving workers. *VLDBJ*, 8(10):1022–1033, 2015.
[9] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*, 2014.
[10] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 workshop on deep learning*, 2014.
[11] L. B. Cornman, R. K. Goodrich, C. S. Morse, and W. L. Ecklund. A fuzzy logic method for improved moment estimation from doppler spectra. *Journal of Atmospheric and Oceanic Technology*, 15(6):1287–1305.
[12] Y. Cui, L. Deng, Y. Zhao, B. Yao, V. W. Zheng, and K. Zheng. Hidden poi ranking with spatial crowdsourcing. In *KDD*, 2019.
[13] H. Dang, T. Nguyen, and H. To. Maximum complex task assignment: Towards tasks correlation in spatial crowdsourcing. In *ICPS*, pages 77–81, 2013.
[14] D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *SIGSPATIAL*, pages 324–333, 2013.
[15] D. Deng, C. Shahabi, and L. Zhu. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *SIGSPATIAL*, page 21, 2015.
[16] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.
[17] A. Finnerty, P. Kucherbaev, S. Tranquillini, and G. Convertino. Keep it simple: Reward and task design in crowdsourcing. In *SIGCHI*, pages 1–4, 2013.
[18] J. Ford, L. R. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 2009.
[19] G. Ghinita, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *VLDBJ*, pages 919–930, 2014.
[20] N. Gulley and J. S. Jang. Fuzzy-logic toolbox. In *Natick: The Math Works Incorporation*, 1995.
[21] S.-W. Huang and W.-T. Fu. Don't hide in the crowd! increasing social transparency between peer workers improves crowdsourcing outcomes. In *SIGCHI*, pages 621–630, 2013.
[22] A. Jameson and B. Smyth. Recommendation to groups. In *Adaptive Web*, page 596, 2007.
[23] L. Kazemi and C. Shahabi. Geocrowd: Enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 189–198, 2012.
[24] L. Kazemi, C. Shahabi, and L. Chen. Geotrucrowd: Trustworthy query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 314–323, 2013.
[25] J. Kleinberg and E. Tardos. Algorithm design. *Prentice Hall*, 2005.

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

[27] C. C. Lee. Fuzzy logic in control systems: Fuzzy logic controller. *IEEE Trans. Syst.*, 20:404–435, 1990.

[28] M. T. Li, L. Xiang, and Y. Jian. Estimating the tolerable waiting time using fuzzy logic. In *CSA*, 2015.

[29] X. Li, Y. Zhao, J. Guo, and K. Zheng. Group task assignment with social impact-based preference in spatial crowdsourcing. In *DASFAA*, 2020.

[30] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*, pages 130–137, 2017.

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

[32] Y. Song and A. T. Johns. Applications of fuzzy logic in power systems. i. general introduction to fuzzy logic. *Power Engineer*, 11(5):219–222, 1997.

[33] G. C. D. Sousa and B. K. Bose. A fuzzy set theory based control of phase-controlled converter dc machine drive. *IEEE Transactions on Industry Applications*, 30(1):34–44, 1994.

[34] M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *Trans. Fuzzy Systems*, 1(1):7, 1993.

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

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

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

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

[39] Y. Tong, Y. Zeng, B. Ding, L. Wang, and L. Chen. Two-sided online micro-task assignment in spatial crowdsourcing. *TKDE*, 2019.

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

[41] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi. Spatial crowdsourcing: a survey. *VLDBJ*, pages 217–250, 2019.

[42] V. V. Vazirani. *Approximation algorithms*. Springer Science and Business Media, 2013.

[43] J. Xia, Y. Zhao, G. Liu, J. Xu, M. Zhang, and K. Zheng. Profit-driven task assignment in spatial crowdsourcing. In *IJCAI*, 2019.

[44] R. Ying, R. He, K. Chen, and P. Eksombatchai. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, 2018.

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

[46] L. A. Zadeh. Fuzzy logic. *Computer*, 21(4):83–93, 2008.

[47] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou. Latency-oriented task completion via spatial crowdsourcing. In *ICDE*, pages 317–328, 2018.

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

[49] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, S. Shang, and K. Zheng. Preference-aware task assignment in spatial crowdsourcing. In *AAAI*, pages 2629–2636, 2019.

[50] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, and X. Zhou. Predictive task assignment in spatial crowdsourcing: A data-driven approach. In *ICDE*, pages 13–24, 2020.

[51] Y. Zhao, K. Zheng, Y. Li, H. Su, J. Liu, and X. Zhou. Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach. *TKDE*, 2019.

Yan Zhao is an Assistant Professor with Aalborg University. She received the Doctoral Degree in Computer Science from Soochow University in 2020. Her research interests include spatial database and trajectory computing.

Kai Zheng is a Professor of Computer Science with University of Electronic Science and Technology of China. He received his PhD degree in Computer Science from The University of Queensland in 2012. He has been working in the area of spatial-temporal databases, uncertain databases, social-media analysis, in-memory computing and blockchain technologies. He has published over 100 papers in prestigious journals and conferences in data management field such as SIGMOD, ICDE, VLDB Journal, ACM Transactions and IEEE Transactions. He is a senior member of IEEE.

Hongzhi Yin received the Ph.D. degree in computer science from Peking University, in 2014. He is a senior lecturer with the University of Queensland. He received the Australian Research Council Discovery Early-Career Researcher Award, in 2015. His research interests include recommendation system, user profiling, topic models, deep learning, social media mining, and location-based services

Guanfeng Liu is current a Lecturer in the Department of Computing at Macquarie University, Australia. He received his Ph.D degree in Computer Science from Macquarie University, Australia in 2013. His research interests include graph data management, trust computing and social networks. He has published over 60 papers in the most prestigious journals and conferences such as IJCAI, AAAI, ICDE, CIKM, TKDE, TSC and ICWS.

Junhua Fang is a lecturer in Advanced Data Analytics Group at School of Computer Science and Technology, Soochow University, Suzhou. Before joining Soochow University, he earned his Ph.D degree in computer science from East China Normal University, Shanghai, in 2017. He is a member of CCF. His research focuses on distributed database and parallel streaming analytics.

Xiaofang Zhou received the bachelor's and master's degrees in computer science from Nanjing University, in 1984 and 1987, respectively, and the PhD degree in computer science from the University of Queensland in 1994. He is a professor of computer science with the University of Queensland. He is the head of the Data and Knowledge Engineering Research Division, School of Information Technology and Electrical Engineering. He is also a specially appointed adjunct professor with Soochow University, China. His research is focused on finding effective and efficient solutions to managing integrating, and analyzing very large amounts of complex data for business and scientific applications. His research interests include spatial and multimedia databases, high performance query processing, web information systems, data mining, and data quality management. He is a fellow of IEEE.