Task Assignment with Worker Churn Prediction in Spatial Crowdsourcing

Ziwei Wang
School of Computer Science and Engineering, University of Electronic Science and Technology of China
ziwei@std.uestc.edu.cn

Yan Zhao*
Department of Computer Science, Aalborg University
yanz@cs.aau.dk

Xuanhao Chen
School of Computer Science and Engineering, University of Electronic Science and Technology of China
xhc@std.uestc.edu.cn

Kai Zheng
School of Computer Science and Engineering, University of Electronic Science and Technology of China
zhengkai@uestc.edu.cn

ABSTRACT

The pervasiveness of GPS-enabled devices and wireless communication technologies flourish the market of Spatial Crowdsourcing (SC), which consists of location-based tasks and requires workers to physically be at specific locations to complete them. In this work, we study the problem of Worker Churn based Task Assignment in SC, where tasks are to be assigned by considering workers' churn. In particular, we aim to achieve the highest total rewards of task assignments based on the worker churn prediction. To solve the problem, we propose a two-phase framework, which consists of a worker churn prediction phase and a task assignment phase. In the first phase, we use an LSTM-based model to extract the latent feelings of workers based on the historical data and then estimate the idle time intervals of workers. In the assignment phase, we design an efficient greedy algorithm and a Kuhn-Munkras (KM)-based algorithm that can achieve the optimal task assignment. Extensive experiments offer insight into the effectiveness and efficiency of the proposed solutions.

KEYWORDS

worker churn, task assignment, spatial crowdsourcing

ACM Reference Format:
Ziwei Wang, Yan Zhao, Xuanhao Chen, and Kai Zheng. 2021. Task Assignment with Worker Churn Prediction in Spatial Crowdsourcing. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482301

1 INTRODUCTION

Crowdsourcing is a computing paradigm, where humans actively or passively participate in the procedure of computing, especially for tasks that are intrinsically easier for humans than for computers [32]. Many successful crowdsourcing platforms exist, e.g., Amazon Mechanical Turk (MTurk)[1] and Wikipedia[2]. Along with the ubiquity of GPS-equipped networked devices such as smartphones, a new class of crowdsourcing, called Spatial Crowdsourcing (SC), has drawn increasing attention in both academia and industry. With SC, requesters can issue spatial tasks (e.g., monitoring traffic conditions or picking up passengers) to SC servers that then assign workers to these tasks (called *task assignment*). The workers complete their tasks by physically moving to the specified locations. Spatio-temporal information (e.g., location, mobility, and the associated contexts) plays a crucial role in SC. Due to its natural connection to the physical world, SC is relevant to a wide spectrum of daily applications, including real-time ride-hailing services (e.g., Uber[3]), and on-wheel meal-ordering services (e.g., GrubHub[4]).

Research on SC [5–8, 32, 34–36, 38–42] has gained momentum in recent years; consequently, many techniques of task assignment are proposed for various application scenarios. For example, Cheng et al. [10] study a reliable diversity-based spatial crowdsourcing (RDB-SC) problem in SC, which aims to maximize the diversity score of assignments. Zhao et al. [37] propose a tensor-decomposition-based algorithm to learn worker preferences, based on which they assign tasks by transforming the assignment problem into a Minimum Cost Maximum Flow (MCMF) problem. The study [17] aims to maximize the number of performed tasks for a worker with an optimal schedule, where they combined two optimization problems: task-matching and task-scheduling.

However, the existing researches focus mainly on spatio-temporal availability of workers and tasks, thus leaving challenges related to effective and efficient task assignment largely unaddressed. For example, the above studies fail to consider user churn (i.e., worker churn) in task assignment, which describes worker defection from an SC service provider. Studies on user churn started from Customer Relation Management, and have been proposed in various service fields [1, 12, 18, 20, 31]. For example, considering user churn as a real and serious business problem, several machine learning methods and artificial neural networks have been proposed to address the problem by telecommunication companies [2, 12, 23]. Besides,

*Corresponding author: Yan Zhao.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482301

[1] https://www.mturk.com/
[2] https://www.wikipedia.com/
[3] https://www.uber.com/
[4] https://get.grubhub.com/

studies on user churn prediction in these fields, such as banks and websites, have been carried out as well [1, 18, 20]. Traditionally, the user churn prediction is treated as a classification problem based on labeled data and feature engineering, where users are generally divided into the churn and non-churn categories [33]. As the SC market saturated due to the globalization of services and fierce competition, the cost of worker acquisition is rising rapidly. Therefore, it is crucial to predict the worker churn and take some measures to retain workers.

In this work, we investigate a task assignment problem in spatial crowdsourcing, called Worker Churn based Task Assignment (WC-TA). To be more specific, given a set of workers and a set of tasks, it aims to achieve the highest total rewards of task assignments based on the worker churn prediction.

In order to tackle the proposed WC-TA problem, we propose a two-phase framework, which includes a worker churn prediction phase and a task assignment phase. The first phase aims to predict the worker churn in the future. More specifically, we capture the latent feelings of workers using an LSTM-based model based on the historical data and then predict the idle time interval of workers, which are compared with a time threshold. In the assignment phase, we propose a greedy and a KM-based method to achieve the task assignment. Specifically, the greedy method aims to assign tasks to workers greedily, and the KM-based method is to find the maximum weight matching on a bipartite graph (composed of workers and tasks) in which the workers that are more likely to be churned are given higher priority.

In summary, our work has four primary contributions:

1) To the best of our knowledge, this is the first work in SC that predicts the worker churn and performs task assignment based on the predictions.

2) We propose an LSTM-based model to extract the workers' latent feelings that are used to learn workers' idle time intervals from the historical task-performing data.

3) We propose greedy and optimal algorithms for achieving the task assignment to trade off the efficiency and effectiveness.

4) Extensive experiments are conducted to verify the efficiency and effectiveness of the proposed methods.

The remainder of this paper is organized as follows. Section 2 introduces the related work and Section 3 provides notations and the proposed problem. In Section 4, a brief introduction of the framework overview is given first, then, we present an LSTM-based model for worker churn prediction and two task assignment algorithms, followed by the experimental results in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

2.1 Task Assignment in Spatial Crowdsourcing

Spatial Crowdsourcing (SC) can be deemed as one of the main enablers to complete location-based tasks [11, 13, 24, 25, 27–30]. According to the task publish mode, SC can be classified into two categories, namely *server assigned tasks* mode (SAT) and *worker selected tasks* mode (WST) [21]. In SAT mode, the server assigns proper tasks to nearby workers based on the system optimization goals, e.g., maximizing the number of assigned tasks [9, 17, 21, 22]. While in WST mode, the server publishes various spatial tasks online, and workers can select any tasks based on their own preferences without the need to coordinate with the server [16, 17].

Most existing studies adopt the SAT mode, where an SC server takes charge of the task assignment process. For example, Cheng et al. [10] propose a reliable diversity-based spatial crowdsourcing (RDB-SC) problem in SC, where an SC server assigns tasks to suitable workers in order to maximize the diversity score of assignments. Zhao et al. [37] propose a preference-based task assignment problem and design a tensor-decomposition-based algorithm to learn worker preferences, based on which they assign tasks by transforming the assignment problem into a Minimum Cost Maximum Flow (MCMF) problem. However, the above studies focus mainly on spatio-temporal availability of workers and tasks, which do not consider user churn (i.e., worker churn) that describes worker defection from an SC service provider.

2.2 User Churn Prediction

User churn prediction is a hot spot in both academia and industry. Traditionally, the problem of user churn prediction is treated as a classification problem, where users are generally divided into the churn and non-churn categories.

In order to solve the problem of user churn prediction, it is necessary to take into account the users' characteristics, including state sequences, behavior sequences, and other features extracted from the historical user profile. Recent studies make great efforts to predict user churn. For instance, Bahnsen et al. [3] introduce a new finance-based approach and develop a cost-sensitive customer churn prediction model, which enables classification algorithms to serve business objectives. Hudaib et al. [19] hybrid a K-means algorithm, Multilayer Perceptron Artificial Neural Networks (MLP-ANN), and Self-Organizing Maps (SOM) to establish a two-stage loss prediction model to predict user churn, where the effectiveness of the solutions is demonstrated on real data. However, due to the data sparsity, the spatio-temporal characteristic, and uncertain churn criteria in SC applications, the aforementioned methods can not be applied to the worker churn in SC directly. In this work, we will combine different models (i.e., LSTM and Fully Connected Neural Network) under different criteria to address the worker churn prediction problem considering SC characteristics.

3 PROBLEM DEFINITION

We proceed to present necessary preliminaries and then define the problem addressed. Table 1 lists the notations used throughout the paper.

Definition 1 (Spatial Task). *A spatial task, denoted by* $s = (l, p, e, r)$*, has a location* $s.l$*, a publication time* $s.p$*, an expiration time* $s.e$*, and a reward* $s.r$*.*

With spatial crowdsourcing, the query of a spatial task s can be answered only if a worker is physically located at that location $s.l$. Besides, considering the expiration time, a spatial task s can be completed only if a worker arrives at $s.l$ before its deadline $s.e$. Note that with the single task assignment mode [21], an SC server should assign each spatial task to only one worker. For simplicity and without loss of generality, we assume the processing time of each task is 0, which means that a worker will go to the next task upon finishing the current task.

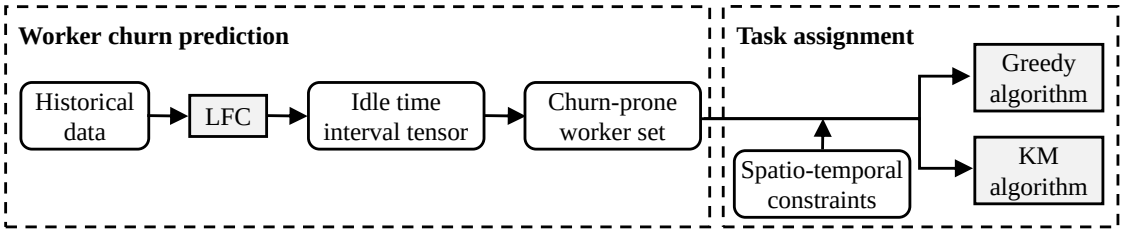


Figure 1: Framework Overview

Table 1: Summary of Notation

Notation	Definition
s	Spatial task
$s.l$	Location of spatial task s
$s.p$	Publication time of spatial task s
$s.e$	Expiration time of spatial task s
$s.r$	Reward of spatial task s
w	Worker
$w.l$	Current location of worker w
$w.d$	Reachable distance of worker w
S_w	A historical task-performing sequence of w
$w.\tau$	Idle time interval of worker w
A	A spatial task assignment
$A.r$	Total reward in task assignment A
$\mathbb{A}$	Spatial task assignment set
μ	Time threshold

Definition 2 (Worker). *A worker, denoted by $w = (l, d)$, has a location $w.l$, and a reachable distance $w.d$. The reachable range of worker w is a circle with $w.l$ as the center and $w.d$ as the radius, within which w can accept assignments.*

A worker can be in either online or offline mode. A worker is online when being ready to accept tasks. In our work, a worker can handle only one task at a certain time instance, which is reasonable in practice. Once the SC server assigns a task to a worker, the worker is considered being offline until the assigned task is completed.

Definition 3 (Idle Time Interval). *Given a worker w who has performed n tasks in a time period, we define the task-performing history of w as a time-ordered task sequence, $S_w = (s_1, s_2, ..., s_n)$. The idle time interval of worker w is the time interval between two adjacent performed tasks, i.e., $w.\tau_i = s_{i+1}.t_s - s_i.t_e$ $(i > 0)$, where $w.\tau_i$ denotes the idle time interval of w between s_i and s_{i+1}, $s_{i+1}.t_s$ denotes the start time (i.e., time of assignment) of s_{i+1}, and $s_i.t_e$ is the completion time of s_i.*

In the rest of the paper, we will use the terms *idle time interval* and *idle interval* interchangeably.

Definition 4 (Worker Churn). *Given a time threshold μ, if a worker w is neither online nor performing tasks within a time period that exceeds μ, worker w is regarded as a churned worker.*

Definition 5 (Spatial Task Assignment). *Given a set of worker $W = \{w_1, w_2, ..., w_{|W|}\}$ and a set of tasks $S = \{s_1, s_2, ..., s_{|S|}\}$, we define A as the spatial task assignment, denoted by A, 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' spatio-temporal constraints.*

We use $A.r$ to denote the total reward in task assignment A. The problem investigated can be stated as follows.

Worker Churn based Task Assignment (WC-TA) Problem Statement. Given a set of online workers W and a set of tasks S at the current time instance on an SC platform, our problem is to find an optimal task assignment A_{opt} that achieves the following goals:

1) primary optimization goal: maximize the total reward among workers, i.e., $\forall A_i \in \mathbb{A}\ (A_{opt}.r \geq A_i.r)$, where $\mathbb{A}$ denotes all possible assignments; and

2) secondary optimization goal: minimize the worker churn rate.

4 ALGORITHM

4.1 Framework Overview

Our framework (cf. Figure 1) is comprised of two components: worker churn prediction and task assignment.

The first component aims to predict workers' churn by calculating the idle time intervals for all the workers based on workers' historical task-performing data. To this end, we utilize a Latent Feeling Capture (LFC) model, which is a regression model that predicts the idle time intervals of workers. More specifically, we extract the latent feeling of each worker from the historical data, including worker ID, reward, time interval and spatial distance between two adjacent tasks, and geographic locations of tasks. Taking the data related to latent feelings of each worker as input, the LFC model generates a worker idle time interval tensor X, where each entry is the prediction of a worker's idle time interval in a certain time slot. Given a time threshold μ, e.g., a month, if the predicted idle time interval of a worker exceeds μ, the worker is judged to be a churn-prone worker, otherwise, the worker is regarded as an active one.

In practice, it is necessary for an SC server to take measures to retain the churn-prone workers to ensure continuous and high worker participation and satisfaction. The measures include assigning the churn-prone workers high-value tasks, such as tasks with high rewards or tasks with good positions. Therefore, in the assignment component, by considering trip constraints including workers' reachable region and tasks' expiration time, we assign tasks to the suitable workers giving high priority to the churn-prone workers. For the sake of efficiency, we propose a Churn-aware Greedy algorithm that tries to assign tasks to the suitable workers who are most likely to churn. We also develop a Churn-aware KM algorithm to maximize the total reward while giving priority to the churn-prone workers when assigning tasks.

4.2 Worker Churn Prediction

In this section, we estimate the idle time intervals for each worker using a Latent Feeling Capturing (LFC) model, based on which we obtain the churn-prone workers by introducing a time threshold.

4.2.1 Behavior-based Modeling for Worker Churn Prediction. Traditionally, the problem of user churn prediction is treated as a classification problem with labeled data [33]. However, in the fields like SC, the standards of worker churn are not always the same and it is hard to tell if a worker is really churned. Besides, we can not get labeled data in SC. To solve these issues, we transform the worker churn problem into a behavior-based problem and introduce a BMM-UCP (Behavior-based Modeling for User Churn Prediction) method [33]. Different from traditional methods of user churn prediction, BMM-UCP converts the classification problem into a regression problem, which predicts the idle time intervals of users. More specifically, given a time threshold μ, BMM-UCP is to train a model M that maximizes the accuracy defined as Equation 1.

$$accuracy = \frac{\sum_i^N l((\tau_i > \mu \wedge \tau_i^* > \mu) \vee (\tau_i \leq \mu \wedge \tau_i^* \leq \mu))}{N} \tag{1}$$

$$l(x) = \begin{cases} 1 & \text{if } x = True \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $l(x)$ is an indication function (cf. Equation 2), τ_i is the predicted idle time interval of worker w_i, τ_i^* is the true value of the time interval, $\wedge$ is logical AND operation, $\vee$ is logical OR operation, and N is the number of workers.

4.2.2 Worker Churn Prediction based on Latent Feeling Capturing (LFC) Model. In this part, we introduce the proposed Latent Feeling Capturing (LFC) model from three aspects, i.e., input, latent feeling capturing, and output, the structure of which is shown in Figure 2.

Input. In this part, we introduce the input of LFC. In practice, it is obvious that workers' idle time intervals are affected by workers' latent feelings such as satisfaction and passion. Therefore, we introduce two kinds of vectors as the input of LFC, i.e., satisfaction-related vector and passion-related vector. For worker w, the satisfaction-related vector, denoted by (r_t, d_t, lo_t, la_t), where r_t represents the reward of the tth task in the historical data of w, d_t represents the geographical distance between the tth task and the $t-1$th task, lo_t represents the longitude of the tth task, and la_t represents the latitude of the tth task. Passion is a feeling similar to satisfaction, which is denoted by (r_t, wt_t, lo_t, la_t), where wt_t represents the time interval between the tth task and the $t-1$th task in the historical data of w, and the remaining three elements (i.e., r_t, lo_t and la_t) have the same meanings as the corresponding elements in satisfaction-related vector. LFC is an LSTM-based model, as a result of which we set the time-step length to 10, i.e., an input instance consists of 10 satisfaction-related vectors and 10 passion-related vectors.

Latent Feeling Capturing. As we discussed above, the idle time interval is related to a worker's latent feelings. It is obvious that a worker's satisfaction/passion after performing the tth task is related to her satisfaction/passion before the tth task and some attributes of the tth task. We regard the latent feelings (e.g., satisfaction and passion) as sequential data, which means that the latent feelings have sequential dependencies. As shown in Figure 2, LFC contains two-way LSTM, where one way is used to capture satisfaction and the other way is used to capture passion.

We will first introduce the process of capturing satisfaction. We take NL as a non-linear translation. Given the historical data of worker w, denoted by $S_w = \{s_1, s_2, ..., s_n\}$, the satisfaction after performing task s_t depends on features of the spatial task s_t and previous satisfaction. So, we can calculate the satisfaction of the current moment by the satisfaction-related vector we have got and previous satisfaction as shown in Equation 3.

$$\mathcal{S}_t = NL_{st}(s_t) + NL_{ss}(\mathcal{S}_{t-1}) \tag{3}$$

where $\mathcal{S}_t$ is worker w's satisfaction after performing the tth task.

The process of calculating passion is similar to that of calculating satisfaction, which is shown in the following.

$$\mathcal{P}_t = NL_{pt}(s_t) + NL_{pp}(\mathcal{P}_{t-1}) \tag{4}$$

where $\mathcal{P}_t$ is worker w's passion after performing the tth task.

As discussed above, both satisfaction and passion have sequential dependencies. LSTM solves the long-term dependency problem through three gate mechanisms and has excellent performance when processing sequential data. So, we introduce LSTM [4] into our model to capture satisfaction and passion.

Next, we illustrate the LSTM-based satisfaction capturing, which is shown in the following equations.

$$f_t^s = \sigma(W_f^s \cdot [H_{t-1}^s, v_t^s] + b_f^s) \tag{5}$$

$$i_t^s = \sigma(W_i^s \cdot [H_{t-1}^s, v_t^s] + b_i^s) \tag{6}$$

$$\tilde{C_t^s} = tanh(W_c^s \cdot [H_{t-1}^s, v_t^s] + b_c^s) \tag{7}$$

$$o_t^s = \sigma(W_o^s \cdot [H_{t-1}^s, v_t^s] + b_o^s) \tag{8}$$

$$\mathcal{S}_t = f_t^s * \mathcal{S}_{t-1} + i_t^s * \tilde{C_t^s} \tag{9}$$

$$H_t^s = o_t^s * tanh(\mathcal{S}_t) \tag{10}$$

where f_t^s indicates what to forget, and the concatenation of satisfaction-related vector v_t^s and filtered satisfaction H_{t-1}^s from the last time step are taken as input. We use a sigmoid function as the activation function when calculating f_t^s. Next, i_t^s indicates what to update, the input and activation function are the same as those of calculating f_t^s but parameters are different. Further, $\tilde{C_t^s}$ is a candidate satisfaction, which is calculated by taking the concatenation of v_t^s and filtered satisfaction H_{t-1}^s as input and selecting a tanh function for activation. Then, the new satisfaction is an addition of the values remembered from $\mathcal{S}_{t-1}$ (calculated by $f_t^s * \mathcal{S}_{t-1}$) and the values that need to be updated from the candidate satisfaction $\tilde{C_t^s}$ (calculated by $i_t^s * \tilde{C_t^s}$). Finally, o_t^s indicates what to output, which is based on our cell state. The filtered satisfaction H_{last}^s of the last time step will be used as the output of LSTM for idle time interval calculation. The structure of the LSTM-based satisfaction capturing is shown in Figure 3.

The process of capturing passion is similar to that of satisfaction, which is omitted due to space limit.

Output. As discussed above, the idle time interval is related to the latent feelings including satisfaction and passion. So, the satisfaction and passion output by the two-way LSTM will be concatenated and then pass through a fully connected layer, by which the predicted idle time intervals are output. The idle time intervals are concatenated into an idle time interval tensor X. Detailed steps are shown in Equations 11 and 12.

$$\tau_w = W_{sp}[\mathcal{S}_w, \mathcal{P}_w] + b_{sp} \tag{11}$$

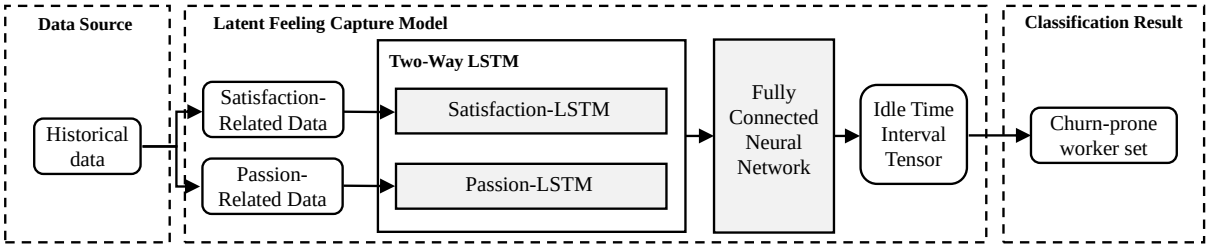


Figure 2: Latent Feeling Capturing Model

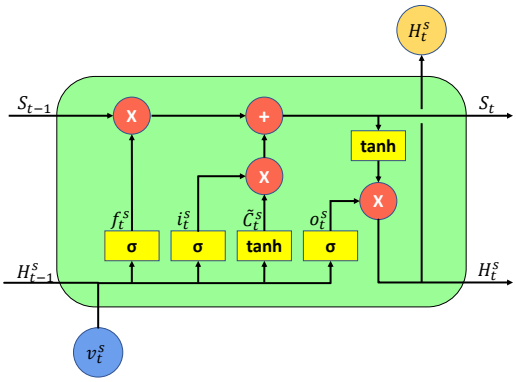


Figure 3: LSTM-based Satisfaction Capturing

$$X = [\tau_1, \tau_2, \tau_3, ...] \tag{12}$$

where τ_w is the predicted idle time interval of worker w, $W_{sp}$is the weight matrix, and b_{sp} is the bias.

Given a time threshold μ, if the corresponding entry of worker w, $X[w]$, exceeds μ, w is judged as a churn-prone worker.

4.3 Task Assignment

In a 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 WC-TA problem. Since an SC server only has local knowledge of the available tasks and workers at any instance of time instead of a global view of all the workers and tasks, we will optimize the task assignment locally at every time instance by maximizing the current assignments and giving higher priorities to workers who are more likely to be churned. We propose two task assignment algorithms on this basis, a Churn-aware Greedy Algorithm and a Churn-aware KM Algorithm.

4.3.1 Churn-aware Greedy Algorithm. Taking workers' idle time intervals as the priority of task assignment, we propose a basic greedy solution to solve the WC-TA problem.

Specifically, given a set of online workers, $W = \{w_1, w_2, ..., w_{|W|}\}$, and a set of available tasks, $S = \{s_1, s_2, ..., s_{|S|}\}$ at the current time, the available workers for spatial task s ($s \in S$), denoted as $AW(s)$, should satisfy the following two conditions: $\forall w \in AW(s), s \in S$,

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

2) $t_{now} + t(w.l, s.l) \leq s.e$,

where $dis(w.l, s.l)$ is the 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 $s.l$. Further, $|AW(s)|$ denotes the number of available workers for spatial task s. For the sake of simplicity, we assume that all the workers share the same speed, so the travel time cost between two locations can be estimated with their Euclidean distance, e.g., $t(w.l, s.l) = dis(w.l, s.l)$. However, our proposed algorithms are not dependent on this assumption and can handle the case where workers are moving at different speeds.

The task assignment algorithm is shown in Algorithm 1. Given a worker set W and a task set S, the LFC model first predicts the idle time intervals of each worker (lines 2–3). Second, Algorithm 1 calculates the set of available workers for each task according to the conditions described above (line 5). Then, we sort the workers in the available worker set for each task in descending order according to workers' idle time interval (line 6). Finally, each task is assigned to the worker with the largest idle time interval (line 7).

Algorithm 1: Churn-aware Greedy Algorithm

Input: W, S
Output: A
1 $A \leftarrow \emptyset$;
2 **for** *each worker* $w \in W$ **do**
3 Predict the idle time interval τ of w;
4 **for** *each task* $s \in S$ **do**
5 $AW(s) \leftarrow$ Find the set of available workers of s;
6 $\tilde{AW}(s) \leftarrow$ Sort $AW(s)$ in descending order of τ;
7 $A \cup [(s, \tilde{AW}(s)[0])]$;
8 **return** A;

4.3.2 Churn-aware KM Algorithm. In this part, we transform the WC-TA problem to a Bipartite Maximum Weight Matching problem and apply the KM algorithm to solve it. The Bipartite Maximum Weight Matching is based on a graph, which is represented by $G = (V, E)$ with V corresponding to the set of vertices and E the set of edges. Given a set of online workers, $W = \{w_1, w_2, ..., w_{|W|}\}$, and a set of available tasks, $S = \{s_1, s_2, ..., s_{|S|}\}$, the number of V and the number of E are fixed to $|W| + |S|$ and $\sum_{i=1}^{|W|} m_i$, respectively, where m_i is the number of worker w_i's adaptive available assignments, which is a subset of worker w_i's available assignments $AS(w_i)$ and is positively related to the predicted idle time interval of w_i. $AS(w)$ should meet the conditions mentioned in Section 4.3.1. For the

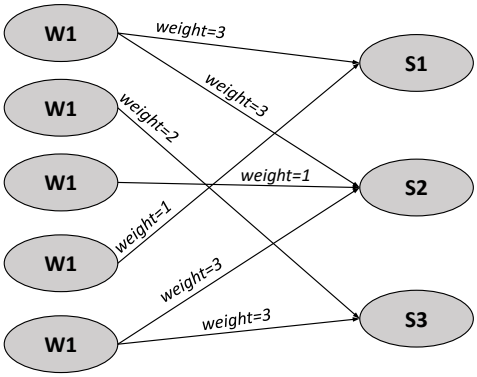


Figure 4: Worker-Task Bipartite Graph

vertices construction, the entire point set V is divided into two sets V_W and V_S, where $V_W \cap V_S = \emptyset$. Each worker w_i maps to a vertex, v_i^W, and each spatial task s_j maps to a vertex, v_j^S.

Due to the spatio-temporal constraints, we add an edge from v_i^W mapped from $w_i \in W$ to the vertex v_j^S mapped from $s_j \in S$ if s_j can be assigned to w_i , i.e., $s_j \in AS(w_i)$. For each edge (v_i^W, v_j^S), its weight (denoted by $weight(v_i^W, v_j^S)$) can be measured as the weighted sum of the time interval τ_i and the reward of the spatial task s_j, i.e., $weight(v_i^W, v_j^S) = w_c * X[i] + w_r * s_j.r$, where w_c is the weight of w_i's predicted idle time interval and w_r is the weight of s_j's reward. Figure 4 depicts an example of a graph for five workers and three tasks.

Algorithm 2: FindTask Algorithm

Input: *worker w, recursion deepth*
Output: *Bool*

```
1  vis_worker[w] = True;
2  if recursion deepth > λ then
3      return False;
4  else
5      for each task s is adjacent to w in G do
6          if vis_task[s] then
7              continue;
8          gap ← ex_worker[w] + ex_task[s] − weight(v_w^W, v_s^S);
9          if gap = 0 then
10             if A[s]=-1 or FindTask(A[s],recursion deepth+1)
                 then
11                 A[s]=w;
12                 return True;
13         else
14             slack[s] = min(slack[s], gap);
15     return False;
```

To improve efficiency, we limit the number of edges in the graph. Each worker node v_i^W has an adaptive upper limit u_i. It means that the number of edges associated with a worker node cannot exceed u_i, which is calculated by $\rho * X[i]$, where ρ is a hyperparameter. For the same consideration, the number of associated edges of each task node cannot exceed u_s, in order to reduce the competition among workers and the recursion depth of the algorithm. u_s is a hyperparameter as well.

The WC-TA problem is now converted into a Bipartite Maximum Matching problem in the direct graph G, which is to achieve the maximum weight matching of G. In our work, we use the KM algorithm with a limit of recursion to find the maximal weight matching.

For better understanding, before introducing the Churn-aware KM algorithm, we will introduce the FindTask algorithm first. FindTask algorithm is a depth first search (DFS) algorithm to find a task for a worker. In the algorithm, we calculate the difference between the weight of the edge associated with the two vertices and the sum of expectations of the worker and the task. If the difference is equal to 0, the task can be assigned to the worker (line 8). If the task has been assigned to another worker, we try to assign another task to that worker (line 9–12). But the depth of recursion cannot exceed the upper recursion limit λ (line 2–3).

Algorithm 3: Churn-aware KM Algorithm

Input: G
Output: A

```
1  A ← [−1, −1, ...];
2  ex_task ← [0, 0, ...];
3  slack ← [INF, INF, ...];
4  for each worker w ∈ W do
5      ex_worker[w] ← max(weight(v_w^W, v_s^S));
6  for each worker w ∈ W do
7      while ex_worker[w] > 0 do
8          vis_task ← [False, False, ...];
9          vis_worker ← [False, False, ...];
10         if FindTask(w,0) then
11             break;
12         else
13             d=INF;
14             for each task s ∈ S do
15                 if ! vis_task[s] then
16                     d=min(d,slack[s]);
17             for each worker w ∈ W do
18                 if vis_worker[w] then
19                     ex_worker[w]− = d;
20             for each task s ∈ S do
21                 if vis_task[s] then
22                     ex_task[s]+ = d;
23                 else
24                     slack[s]− = d;
25 return A;
```

The KM task assignment algorithm is shown in Algorithm 3. Given the bipartite graph G, which is composed of two vertices sets V_S and V_W. First, for each vertex in V_W, its expectation is equal to the largest weight among the edges associated with it in graph G (lines 4–5). Second, Algorithm 3 recursively finds matching tasks for

worker w through the FindTask function (line 10). Third, if w fails to match a task, we adjust the expectations of workers and tasks involved in the last matching to change the competitive relationship among workers so that more workers can be assigned (lines 12–24).

The original KM algorithm is used to find the perfect matching of a weighted bipartite graph. However, considering that a perfect matching may not exist in a worker-task bipartite graph, we propose some optimization strategies to improve the KM algorithm. In the original KM algorithm, it will not stop trying to match tasks for a worker until a successful match, which may cause an endless loop in our problem. Therefore, in our algorithm, if the expectation of w is less than 0, we stop matching tasks for the worker (line 7).

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

We use a check-in dataset from Yelp to simulate our problem, which is a common practice in evaluation of SC platforms [9, 15, 16]. In order to make the user data more representative, we filter the data, where we only use the data of users with number of reviews exceeding 20 and the number of reviews before 2019-05-14 23:22:59 exceeding 10. The resulting dataset provides check-in data from 8 metropolitan areas in the USA, which includes 160,585 POIs and 31,262 users. We assume that we assign tasks to workers in a certain time slot in the near future i.e., 2019-05-14 23:22:59. For our experiments, we assume that the users are the workers of SC systems since users who check in to different spots are good candidates to perform spatial tasks in the vicinity of those spots, and their locations are those of the most recent check-in points. We assume that all users in the testing set will be online in that time slot. For each POIs, we use its location and stars as the location and reward of a task, respectively. Checking in a POI is equivalent to accepting a task. The distance is calculated by the Euclidian distance. The values of all parameters used in our experiments are summarized in Table 2, where the default values of all parameters are underlined. All the algorithms are implemented on an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, and NVidia GeForce RTX 2080Ti GPU.

Table 2: Experimental Parameters

Parameter	Values
Number of tasks $\|S\|$	1000, <u>2000</u>, 3000, 4000, 5000
Number of workers $\|W\|$	1000, 2000 , 3000, 4000, <u>5000</u>
Reachable distance of workers d	0.05, 0.1, <u>0.5</u>, 1.0, 5.0
Valid time of tasks $e-p$	0.05, 0.1, <u>0.3</u>, 0.5, 0.7
Limit coefficient k	2.0, 4.0, <u>6.0</u>, 8.0, 10.0

5.2 Experiment Results

5.2.1 Performance of Worker Churn Modeling. We first evaluate the performance of worker churn prediction.

To evaluate the accuracy of worker churn prediction, we adopt the metric, *accuracy*, shown in Equation 1. We randomly remove 20% of the reviews from the review data set, which are used as the testing set to evaluate the inferred values and will be used as simulation data for the experiments of the task assignment phase. The remaining 80% are used as the training data.

Table 3: Accuracy of Worker Churn Prediction

Methods	Time threshold			
	half a month	one month	two months	three months
LR	37.91	57.16	77.66	84.08
MC	36.89	57.23	77.89	83.99
LFC	**61.37**	**67.97**	**78.92**	**84.72**

Two baseline algorithms are introduced to compare with our LFC method, a linear regression (LR) algorithm [26] and a multi-layer fully connected neural network (MC) algorithm [14]. The linear regression algorithm is a statistical analysis method that uses regression analysis in mathematical statistics to determine the quantitative relationship between two or more variables. Its expression is $y = wx + b$, where x is historical data, and y denotes the predicted idle time interval in our problem. The multi-layer fully connected neural network is a computing model, which is composed of a large number of nodes (or neurons) connected to each other. Each node represents a specific output function, called an activation function. Each connection between two nodes represents a weighted value for the signal passing through the connection, called a weight, which is equivalent to the memory of an artificial neural network. The output of the network is different depending on the connection method of the network, the weight value and the activation function. The network itself is usually an approximation of a certain algorithm or function in nature, or it may be an expression of a logic strategy.

Table 3 shows the evaluation results. When the time threshold μ is set to half a month, one month, two months and three months, LFC always performs the best among the methods. This demonstrates the superiority of LFC for predicting the worker churn.

5.2.2 Performance of Task Assignment. For the performance of task assignment, we study the following algorithms.

1) Greedy: The greedy task assignment algorithm that does not consider the worker churn.

2) KM: The KM task assignment algorithm that does not consider the worker churn.

3) Greedy+WC: The greedy task assignment algorithm based on the worker churn predicted by LFC.

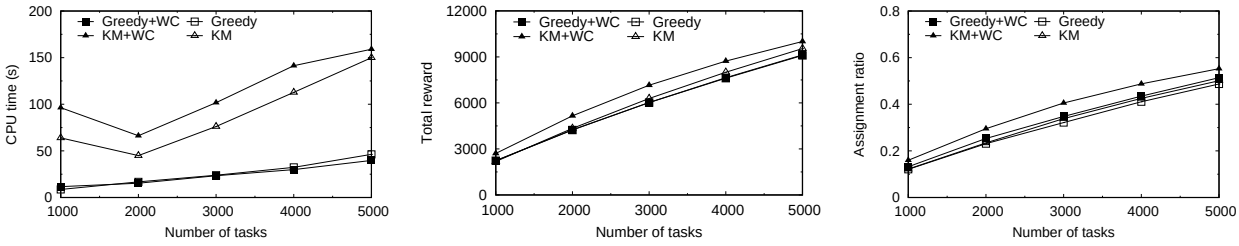
4) KM+WC: The KM task assignment algorithm based on the worker churn predicted by LFC.

Three metrics are compared among the methods: 1) CPU time: the CPU time cost for finding the task assignment; 2) total reward; 3) assignment ratio of churn-prone workers (marked by AR): proportion of workers who are prone to churn that are assigned to tasks, i.e.,

$$AR = \frac{N(W_{assign} \cap W_{churn})}{N(W_{churn})} \tag{13}$$

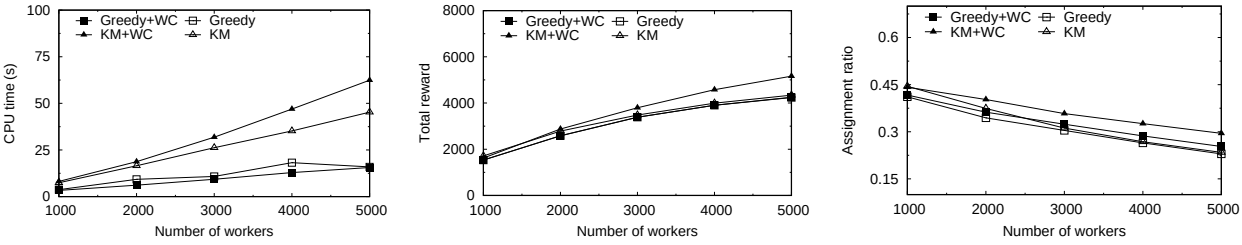
where $N(W_{assign} \cap W_{churn})$ denotes the number of the assigned churn-prone workers, and $N(W_{churn})$ denotes the number of the churn-prone workers.

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 Yelp dataset. As shown in Figure 5(a), for Greedy and Greedy+WC, the CPU time increases as $|S|$ increases, but for KM-based algorithms, the CPU time is not simply positively correlated with $|S|$. When $|S|$ is small, the competition among workers is



(a) CPU time (b) Total reward (c) Assignment ratio

Figure 5: Performance of Task Assignment: Effect of $|S|$



(a) CPU time (b) Total reward (c) Assignment ratio

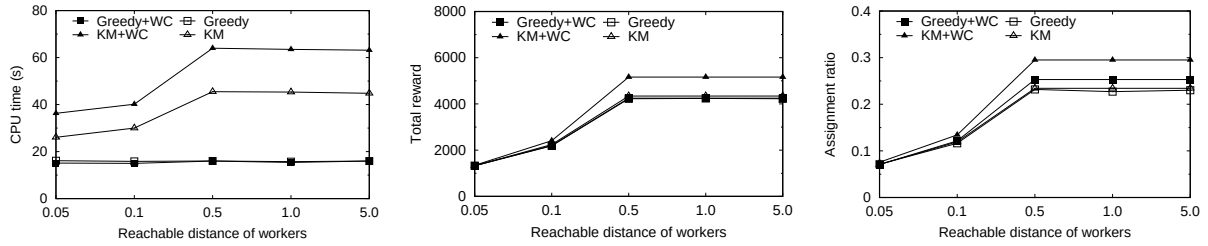
Figure 6: Performance of Task Assignment: Effect of $|W|$

intense, which may increase the recursion depth of the algorithms (e.g., KM and KM+WC). For example, the algorithm wants to assign task s_1 to worker w_1, but s_1 has been assigned to w_2, so the algorithm will try to assign task s_2 to w_2. Due to the small number of tasks, s_2 may also have been assigned to another worker. In this case, the recursion depth will be large. Figure 5(b) shows the total reward and as the number of tasks increases, the total rewards obtained by the four algorithms increase. Since the greedy algorithm is an algorithm that only focuses on the local optimal solution, its performance is always worse than those KM-based algorithms (including KM and KM+WC). However, in the bipartite graph of KM+WC, churn-prone workers can be associated with more task vertices, which changes the competitive relationship among workers. So, the reward of KM+WC is more than that of the original KM algorithm. At the same time, as depicted in Figure 5(c), KM+WC and Greedy+WC have better performances in assignment ratio (AR). As the number of tasks increases, more workers can be assigned to tasks, so the AR of the four algorithms increase. Since Greedy+WC and KM+WC consider worker churn and give higher priority to churn-prone workers, they have greater assignment ratios. Especially for KM+WC, its assignment ratio is about $5\% - 20\%$ higher than those of other baseline algorithms.

Effect of $|W|$. To study the effect of $|W|$, we generate 5 datasets containing 1000 to 5000 workers by random selection from the testing set. As depicted in Figures 6(a) and 6(b), as $|W|$ increases, more tasks can be assigned to workers and more workers can get tasks, so the total rewards increase. At the same time, in KM and KM+WC, as $|W|$ increases, the competition among workers becomes more intense. That means the algorithms need to recurse multiple times when assigning tasks to them. At the same time, although $|W|$ increases, the number of tasks does not change, and some tasks cannot be assigned to workers due to the constraints of workers and tasks. Therefore, the AR of the four algorithms drops, which is shown in Figure 6(c). But KM+WC and Greedy+WC still have better assignment ratios than other methods. For example, the assignment ratio of KM+WC is about $10\% - 20\%$ higher than those of other baseline algorithms, which shows the superiority of our proposed algorithm.

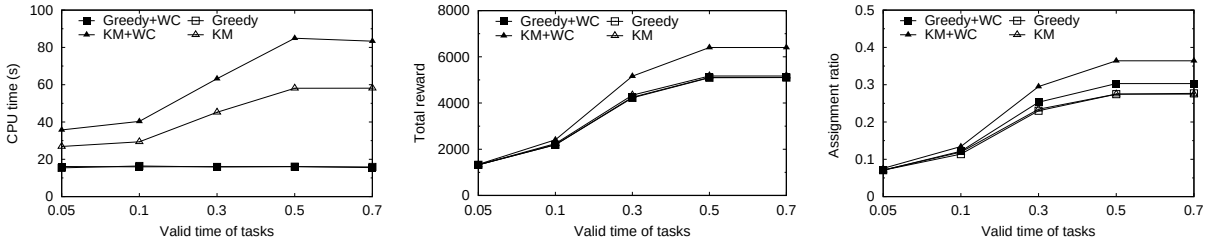
Effect of d. We also study the effect of workers' reachable distances d by changing it from 0.05 km to 5 km. From Figure 7(a) we can see that, the CPU times of KM and KM+WC increase faster than those of Greedy and Greedy+WC as d increases. This is because that as d increases, there are more available tasks for each worker and the competition among workers will be more intense, which will increase the recursion depth of algorithms. Moreover, workers with larger reachable distances tend to have more available task assignments, which leads to more edges in the graphs of KM and KM+WC. As a result, the AR and total reward increase as d increases, which is shown in Figures 7(c) and 7(b). However, limited by the number of tasks and workers, as d increases, its effect on algorithms tends to be saturated. As shown in Figure 7, all the four methods remain unchanged after d exceeds 0.5km.

Effect of $e - p$. We then study the effect of the valid time $e - p$ of tasks. In Figure 8, when $e - p$ increases, tasks can be assigned to more workers. Each worker has a greater probability of being assigned to tasks, so the total reward and assignment ratio of churn-prone workers increase. As depicted in Figure 8(a), the CPU time of all methods increase. The CPU time of KM and KM+WC algorithms increase faster, with the similar reason of the effect of d. Similar to d, limited by $|S|$ and $|W|$, all the four methods remain unchanged after $e - p$ exceeds 0.5h.



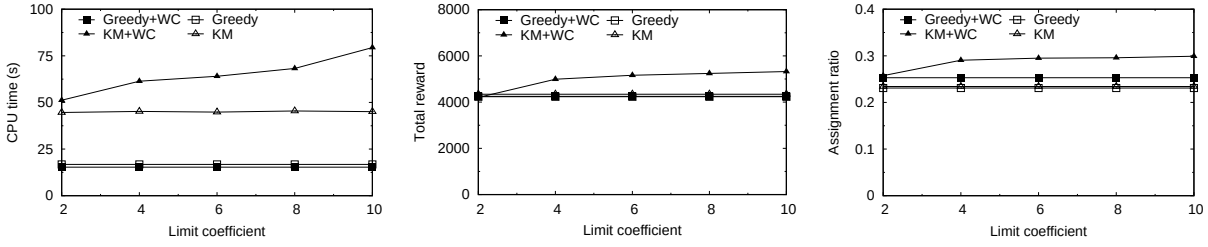
(a) CPU time (b) Total reward (c) Assignment ratio

Figure 7: Performance of Task Assignment: Effect of d



(a) CPU time (b) Total reward (c) Assignment ratio

Figure 8: Performance of Task Assignment: Effect of $e - p$



(a) CPU time (b) Total reward (c) Assignment ratio

Figure 9: Performance of Task Assignment: Effect of k

Effect of k. Finally, we study the effect of k, which limits the number of edges associated with worker vertices in KM+WC algorithm. As k increases, in the graph of KM+WC, each worker can be associated with more tasks, in this way, workers have more opportunities to be assigned to tasks, but the competition among workers is more intense as well, leading to an increase in CPU time, which is shown in Figure 9(a). But at the same time, as k increases, KM can find a matching with greater weights. So the assignment ratio of churn-prone workers and total reward increase, which is shown in Figure 9(b) and 9(c). k is a parameter only for KM+WC. The curves of other algorithms in Figure 9 are the results with default values, in order to show the effect of k better. Experimental results show that, our proposed KM+WC algorithm does have superior performances and can perform better as k increases.

6 CONCLUSION AND FUTURE WORK

The ubiquity of mobile devices with high-fidelity sensors and the sharp decreases in the cost of ultra-broadband wireless networks flourish the market of Spatial Crowdsourcing (SC), which consists of location-specific tasks and requires workers to physically be at specific locations to complete them. In this paper, we study a novel task assignment problem in SC, namely Worker Churn based Task Assignment (WC-TA). We address a few challenges by proposing different strategies to identify the workers who are easy to churn, and consider their feelings when assigning tasks, so that they can get a better experience on SC platforms. To the best of our knowledge, this is the first work in SC that predicts the worker churn and performs task assignment based on the predictions. Extensive experiments demonstrate the effectiveness of our proposed solutions.

ACKNOWLEDGMENTS

This work is partially supported by Natural Science Foundation of China (No. 61972069, 61836007 and 61832017).

REFERENCES

[1] Jaehuyn Ahn, Junsik Hwang, Doyoung Kim, Hyukgeun Choi, and Shinjin Kang. A survey on churn analysis in various business domains. *Access*, 8:220816–220839, 2020.

[2] Adnan Amin, F. Al-Obeidat, B. Shah, May Al Tae, C. Khan, Hamood Ur Rehman Durrani, and S. Anwar. Just-in-time customer churn prediction in the telecommunication sector. *J SUPERCOMPUT*, 76:3924–3948, 2017.

[3] Alejandro Correa Bahnsen, Djamila Aouada, and B. Ottersten. A novel cost-sensitive framework for customer churn predictive modeling. *Decision Analytics*, 2:1–15, 2015.

[4] Yoshua Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *T NEURAL NETWOR*, 52:157–66, 1994.

[5] Z. Chen, P. Cheng, Y. Zeng, and L. Chen. Minimizing maximum delay of task assignment in spatial crowdsourcing. *ICDE*, pages 1454–1465, 2019.

[6] Z. Chen, Peng Cheng, Liquan Chen, Xuemin Lin, and C. Shahabi. Fair task assignment in spatial crowdsourcing. *PVLDB*, 13:2479 – 2492, 2020.

[7] P. Cheng, L. Chen, and J. Ye. Cooperation-aware task assignment in spatial crowdsourcing. *ICDE*, pages 1442–1453, 2019.

[8] P. Cheng, Xiang Lian, Xun Jian, and Lei Chen. Frog: A fast and reliable crowdsourcing framework. *TKDE*, 31:894–908, 2019.

[9] Peng Cheng, Xiang Lian, Lei Chen, and Cyrus Shahabi. Prediction-based task assignment in spatial crowdsourcing. In *ICDE*, pages 997–1008, 2017.

[10] Peng Cheng, Xiang Lian, Z. Chen, L. Chen, J. Han, and J. Zhao. Reliable diversity-based spatial crowdsourcing by moving workers. *PVLDB*, 8:1022–1033, 2015.

[11] Y. Cheng, Boyang Li, Xiangmin Zhou, Y. Yuan, G. Wang, and L. Chen. Real-time cross online matching in spatial crowdsourcing. *ICDE*, pages 1–12, 2020.

[12] Alae Chouiekh and E. Haj. Deep convolutional neural networks for customer churn prediction analysis. *Int. J. Cogn. Informatics Nat. Intell.*, 14:1–16, 2020.

[13] Yue Cui, Liwei Deng, Yan Zhao, Bin Yao, Vincent W Zheng, and Kai Zheng. Hidden poi ranking with spatial crowdsourcing. In *KDD*.

[14] Y. Le Cun, L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, and W. Hubbard. Handwritten digit recognition: applications of neural network chips and automatic learning. *COMMUN MAG*, 27:41–46, 1989.

[15] H. Dang, Tuan A. Nguyen, and Hien To. Maximum complex task assignment: Towards tasks correlation in spatial crowdsourcing. In *IIWAS '13*, page 77–81, 2013.

[16] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *SIGSPATIAL*, pages 324–333, 2013.

[17] Dingxiong Deng, Cyrus Shahabi, and Linhong Zhu. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *SIGSPATIAL*, pages 1–10, 2015.

[18] J. Dias, P. Godinho, and Pedro Torres. Machine learning for customer churn prediction in retail banking. *ICCSA*, 12251:576 – 589, 2020.

[19] A. Hudaib, Reham Dannoun, Osama Harfoushi, Ruba Obiedat, and Hossam Faris. Hybrid data mining models for predicting customer churn. *IJCNS*, 08:91–96, 2015.

[20] Marcel Karnstedt, Matthew Rowe, Jeffrey Chan, H. Alani, and Conor Hayes. The effect of user features on churn in social networks. In *WebSci '11*, 2011.

[21] Leyla Kazemi and Cyrus Shahabi. Geocrowd:enabling query answering with spatial crowdsourcing. In *International Conference on Advances in Geographic Information Systems*, pages 189–198, 2012.

[22] Leyla Kazemi, Cyrus Shahabi, and Lei Chen. Geotrucrowd:trustworthy query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 314–323, 2013.

[23] Stefan M. Kostic, Mirjana Simic, and Miroljub V. Kostic. Social network analysis and churn prediction in telecommunications using graph theory. *Entropy*, 22, 2020.

[24] Xiang Li, Yan Zhao, Jiannan Guo, and Kai Zheng. Group task assignment with social impact-based preference in spatial crowdsourcing. In *DASFAA*, pages 677–693, 2020.

[25] Xiang Li, Yan Zhao, Xiaofang Zhou, and Kai Zheng. Consensus-based group task assignment with social impact in spatial crowdsourcing. *Data Science and Engineering*, 5(4):375–390, 2020.

[26] D. Montgomery and Elizabeth A. Peck. Introduction to linear regression analysis. 2001.

[27] Wangze Ni, P. Cheng, L. Chen, and Xuemin Lin. Task allocation in dependency-aware spatial crowdsourcing. *ICDE*, pages 985–996, 2020.

[28] Tianshu Song, Ke Xu, Jiangneng Li, Yiming Li, and Yongxin Tong. Multi-skill aware task assignment in real-time spatial crowdsourcing. *GeoInformatica*, 24:153–173, 2019.

[29] Qian Tao, Yongxin Tong, Zimu Zhou, Yexuan Shi, L. Chen, and K. Xu. Differentially private online task assignment in spatial crowdsourcing: A tree-based approach. *ICDE*, pages 517–528, 2020.

[30] Hien To, C. Shahabi, and Li Xiong. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. *ICDE*, pages 833–844, 2018.

[31] Yongxin Tong, Yu xiang Zeng, Bolin Ding, L. Wang, and L. Chen. Two-sided online micro-task assignment in spatial crowdsourcing. *Transactions on Knowledge and Data Engineering*, 33:2295–2309, 2021.

[32] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, L. Chen, and C. Shahabi. Spatial crowdsourcing: a survey. *VLDB J*, 29:217–250, 2019.

[33] Meng Xi, Zhiling Luo, N. Wang, and Jianwei Yin. A latent feelings-aware rnn model for user churn prediction with behavioral data. *ArXiv*, abs/1911.02224, 2019.

[34] Jinfu Xia, Yan Zhao, Guanfeng Liu, Jiajie Xu, Min Zhang, and Kai Zheng. Profit-driven task assignment in spatial crowdsourcing. In *IJCAI*, pages 1914–1920, 2019.

[35] Yan Zhao, Jiannan Guo, Xuanhao Chen, Jianye Hao, Xiaofang Zhou, and Kai Zheng. Coalition-based task assignment in spatial crowdsourcing. In *ICDE*, pages 241–252, 2021.

[36] Yan Zhao, Yang Li, Yu Wang, Han Su, and Kai Zheng. Destination-aware task assignment in spatial crowdsourcing. In *CIKM*, pages 297–306, 2017.

[37] Yan Zhao, J. Xia, G. Liu, Han Su, Defu Lian, Shuo Shang, and Kai Zheng. Preference-aware task assignment in spatial crowdsourcing. In *AAAI*, pages 2629–2636, 2019.

[38] Yan Zhao, Kai Zheng, Yue Cui, Han Su, Feida Zhu, and Xiaofang Zhou. Predictive task assignment in spatial crowdsourcing: a data-driven approach. In *ICDE*, pages 13–24, 2020.

[39] Yan Zhao, Kai Zheng, Jiannan Guo, Bin Yang, Torben Bach Pedersen, and Christian S Jensen. Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches. In *ICDE*, pages 265–276, 2021.

[40] Yan Zhao, Kai Zheng, Yang Li, Han Su, Jiajun Liu, and Xiaofang Zhou. Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach. *TKDE*, pages 2336–2350, 2019.

[41] Yan Zhao, Kai Zheng, Hongzhi Yin, Guanfeng Liu, Junhua Fang, and Xiaofang Zhou. Preference-aware task assignment in spatial crowdsourcing: from individuals to groups. *TKDE*, 2020.

[42] Libin Zheng and Lei Chen. Multi-campaign oriented spatial crowdsourcing. *TKDE*, 32:700–713, 2020.